

75p

May 1983

ELECTRONICS & COMPUTING

MONTHLY

Britain's First Electronics & Computer Applications Magazine

The
Computer
Brain

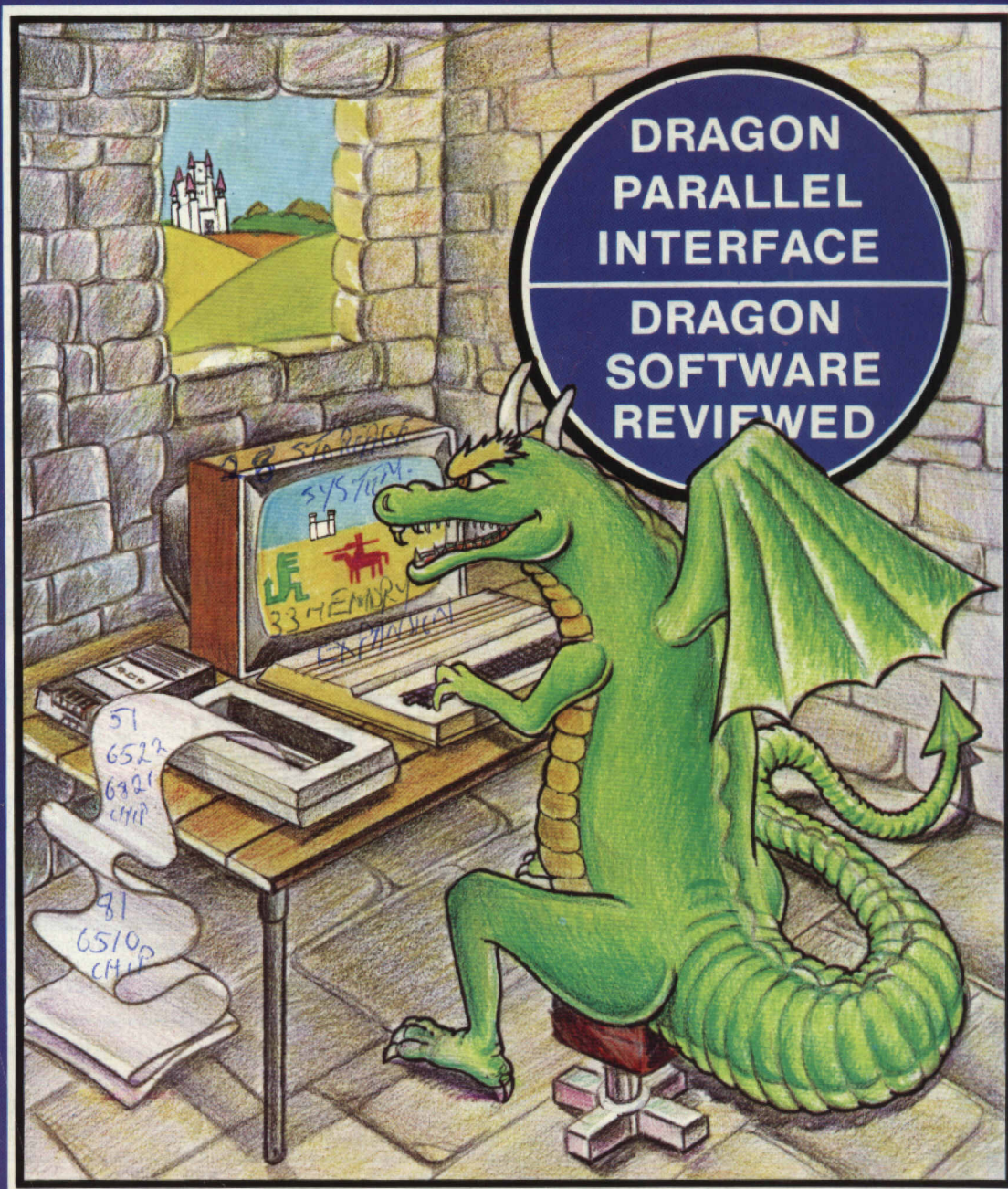
BBC Micro
Experiments

Jupiter Ace
Electronic
Scrambler

Commodore
64 Review

High Density
Floppy
Discs

SPECTRUM RS232
PRINTER
INTERFACE



ELECTRONICS & COMPUTING MONTHLY

40/42 Oxford Street
Daventry
Northampton
NN11 4AD
(03272) 71841/71702

Publisher/Editor
David Raven

Advertising Manager
Claire Fullerton

Deputy Advertising
Manager
Alun Evans

Production
Terry Gibbins

Editorial Assistant
Ann Houghton

Office Manager
Ruby Jordon

Distribution
Circulation Department
EMAP National
Publications Ltd
Bretton Court
Bretton
Peterborough
0733-264666

Published By
EMAP Business &
Computer Publications
Bushfield House
Orton Centre
Peterborough
0733-237111

Publishing Director
Peter Peskett

Production By
Time Graphics
Northampton

Printed By
L.S.G. Lincoln

ABC

MEMBER OF THE AUDIO
BUREAU OF CIRCULATION

High Density Floppy Discs 18

This investigation into the world of the floppy disc reveals an ever-changing technology incorporating some very advanced scientific techniques.

Understanding Digital Electronics Part 3 23

This month Sequential Logic is the topic for discussion leading us further into the building blocks which make up computing circuitry.

The "Soloload" Data Storage System 28

This article started as a product review, however, we felt it was such an innovation that it deserved treating as a technical feature in its own right.

Review of the Basicare System 33

Stephen Adams examined the modular system developed by Basicare which also has considerable technical merit.

The Computer Brain 37

A practical BASIC introduction to intelligent programs. Readers who saw a recent Horizon program which included A.I. (artificial intelligence) will appreciate this series.

Computing Explained Part 3 45

Graphics and colour.

Interfacing the Dragon 51

It's time to get the Dragon hooked up to the outside world. This article describes a simple parallel interface using a 6522VIA.

Spectrum Printer Interface 64

At long last we're able to unveil the Spectrum RS232 interface which some would say is now well overdue. This project opens the door to connecting up the Spectrum to the outside world, and getting a decent printer attached.

Rules for Dialogue Programming 70

Dr. Philip Barker of Teesside Polytechnic describes how to write programs using PILOT.

The BBC Micro and Fourier Synthesis 78

Read news and views this month and you will get the gist of applications for this particular project.

Commodore 64 Review 81

Mike James takes a look at this latest new computer from the makers of the VIC 20.

Dragon Software Review 86

Which, where, what, why... look before you buy is the growing message for software consumers.

Software for the E & CM Computer Project 91

Comment	3
Next Month	3
Readers Letters	7
News & Views	12
New Computer Products	14

Electronics & Computing Monthly is normally published on the 13th day of each month

© copyright EMAP Business & Computer Publications Limited 1983. Reasonable care is taken to avoid errors in this magazine however, no liability is accepted for any mistakes which may occur. No material in this publication may be reproduced in any way without the written consent of the publishers. Subscription rates: UK £9.50 incl. post. For overseas rates apply to Subscription Dept., Competition House, Farndon Road, Market Harborough, Leics.

COMMENT

The quality and complexity of the electronic and computing projects that are now being offered to E & CM, underlines the progress which has been made by computer enthusiasts during the past two years. It would be nice to think that this is partly as a result of E & CM's original concept of 'taking the lid off computers', and getting rid of the 'black box' concept promoted by many computer journals.

The long awaited RS232 interface for the Spectrum, promised not long after the machine's launch, has finally been produced. Not, however, by Sinclair Research or the multitude of commercial companies producing add-on equipment, but by an E & CM reader. In the April issue we showed you how to turn the ZX81 into possibly one of the best cost-saving gadgets yet produced, by using it for home energy management. This very clever piece of work was carried out by a professional engineer who installs multi-thousand pound computer controlled heating systems in hospitals and factories. In his spare time he decided to do the same for the ZX81.

Next month we introduce video into computing with a project which interfaces a video camera to the ZX Spectrum. It's all a long way from two years ago when we were advising you how to "Choose a Home Computer" and "Convert your Digital Clock to a time Switch".

E & CM was never intended as a first time users magazine and it makes no apologies for the occasional high level and academic nature of some of its articles. Artificial Intelligence is coming, and we will need to know about integrated optics and major new developments in memory storage systems.

It's important that electronics and computing remains fun too, since while many of our readers are professional electronics and computing engineers, they also like to relax with their combined hobby and career.

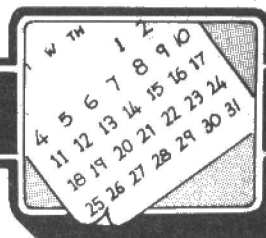
I'm certain we haven't yet got it all completely right, however, since the readership continues to grow we must be on the right track. On major new factor which should help E & CM achieve its goals is the appointment of a new editor starting with the June issue. Gary Evans has recently joined us after a very successful period editing Radio & Electronic World. Gary has a strong background in the electronic and computing specialist press, having previously worked on both Electronics Today and Computing Today. This move should substantially strengthen our editorial team.

Dave Ransley
PUBLISHER

Micronet 800

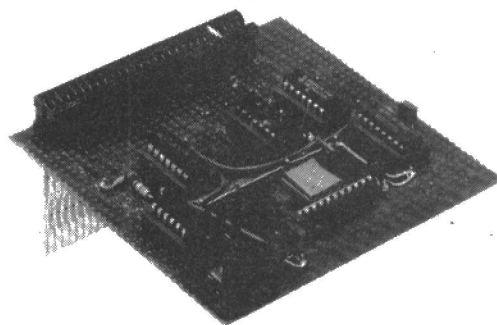
In the April issue I gave an incorrect telephone number for further information on Micronet. This should be 01 837 3699. I've also been asked to point out that the cost is £52 a year plus an additional charge for the telephone modem which starts at £49 plus VAT. Sorry for any confusion which resulted.

NEXT MONTH



Next Month

Spectrum Monitor –
Add an EPROM to your 48K Spectrum.



Spectrum Monitor

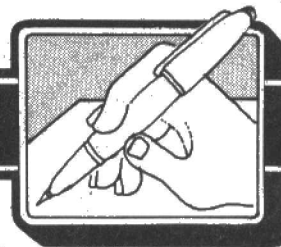
Power Control for Micros –

Three different types of power control units to use with your micro.

Modelling a Rotary Combustion Engine using the ZX81.

Plus regular features
– Computers Explained
and Understanding
Digital Electronics.

(Two of the above features, Power Control for Micros and Modelling a Rotary Combustion Engine were announced for the May issue and held over due to lack of space).



READERS LETTERS

Dear Sir,

I have been interested in hardware add-ons for the Sinclair machines since I purchased a ZX80 and then a ZX81. I discovered your magazine about six months ago and have been very interested in the articles that you have printed. Although I feel that all too often detail is missing and that readers are afraid to take the "leap" into making add-ons because detail is not sufficient.

An example I quote is the A to D convertor for the ZX81. Some people who otherwise might have made the project were baffled because of the lack of detailed specific explanation. If all the pin connections had been made clear instead of just leaving the diagrams as logic gate symbols with only a brief mention of the chips required more people might have been able to undertake the project.

I am interested in developing a series around the ZX81/Spectrum. One idea was to write a built on series. Involving building an input output port with an A to D convertor and then developing it with very simple add ons for: Temperature sensing. Light level sensing. Operating relays, Switching mains voltages. Sound generation and possible sound detection. I have already built circuits for all of the above except sound detection and I am working on that at present (although I have got a crude system working).

I have found that for your A to D convertor the actual chip was almost impossible to obtain. I could only track it down by contacting the manufacturers Mostek International. Frequently for a project parts from many sources are required. So I would be interested in providing a kit of parts or individual/specialist components for my projects.

I look forward to hearing your comments, views, suggestions and any "offers" soon.

Yours sincerely,
Paul J. Smale
80 Hayes Lane,
Bromley, Kent
BR2 9EE

This all sounds like a good idea Paul. Thanks for your constructive criticism, we're obviously striving to achieve higher standards each month and it's offers of help like yours by which we shall achieve them.

Dear Sir,

I have just spent all afternoon trying to work out why the circuit for eliminating ROM echoes on the ZX81 (circuit card no. 8 free with Feb. '83 E & CM doesn't work!).

The circuit's simple enough, and I'm no newcomer to electronics. I've been at it since the age of 14 and hold amateur radio licence G4BPY. I have only been into this computing business for about a month, however. The problem with the circuit, on observing the o/p waveform on an oscilloscope, appears to be that the CMOS chip is unable to handle the high frequency pulses involved, and they were getting very distorted as a result. I tried three 4001's with identical results.

I have enclosed an alternative circuit which does work, using a 74LS00 TTL IC.

Finally, I wonder if anyone could help me obtain a circuit diagram and manual for the "Science of Cambridge" Mk. 14 MPU, which is no longer made.

I know that it used an INS8060 chip, and I have a couple of these I would like to experiment with.

Yours faithfully,
G. M. Pheasant,
Great Wyrley,
Walsall.

Dear Sir,

I refer to your article in the February edition about upgrading of Spectrum RAM.

I opened up my new Spectrum on the day of receipt to see if I could attempt the modification. However, Step 7 involving the removal of the existing memories seems to be an obstacle. In my set all the chips are soldered in and do not appear to be as easily removed as suggested in the article without great risk of damage. To add to my reluctance to 'have a go' I am unable to identify the IC's with any advertised in the magazine.

Is there something missing or has the 'gap' been plugged by the manufacturer?

Yours faithfully,
James Wastin
18 Arleston Road,
Omagh,
Co. Tyrone.

The answer would seem to be yes James. Oh well, we got away with it for a while.

Dear Sir,

Mr. D. Davidson's comments on my Spectrum EPROM Programmer (letters April '83) raise an interesting point. Whilst it is true that to generate a Wait state for more than about 4ms in a normal Z80A system may cause loss of data in dynamic RAMs, this does not necessarily apply to the Spectrum which is rather a special case in having a second controller in the form of the ULA IC. Although I do not fully understand the operations performed by the ULA it clearly has an overriding control of the RAM because it can continue to read from the display file in RAM and produce the TV display even if the Wait line is held permanently low. It seems that this reading provides sufficient activity to keep the RAM refreshed, or perhaps the ULA is designed to interperse refreshes with its display function; either way experiment has shown that data is not lost.

If however a long Wait state is inserted whilst the microprocessor is working from the lower 16K of RAM, part of which contains the display file, then the display generation is inhibited, the display goes black and the system crashes due to data corruption. In my software for the RS232 Interface (May '83) I was careful to avoid this by calling a convenient ROM routine at the point in the code where a Wait could occur. As the POKE instruction that causes the Wait state in the EPROM Programmer is a ROM routine anyway no special precautions are necessary.

Yours faithfully,
J. Williams
Watford,
Herts.

Dear Editor,

Birmingham Nascom User Group

Would you please note that we meet on the last Thursday of every month (except December) at 8.00 p.m. in the upstairs room at Davenports Social Club, Granville Street, Birmingham (behind the Brewery, off Bath Row, near the Birmingham Accident Hospital).

The Club Secretary, Martin Sidebottom, can be contacted at home on 021 744 3093, for further details.



ⓧ New Applications for Personal Computers

Going virtually unnoticed, except perhaps by the dedicated electronics enthusiast, is yet another fast moving product trend which – as is the way nowadays – started in California, and will soon be over here.

Turning your personal computer into a top notch, professional standard, test or measuring instrument by using relatively simple and low cost plug-on modules plus some operating software, is an idea which is catching on rapidly. It offers exciting possibilities to cash-tight schools, colleges and indeed anyone who cannot justify to his company (or wife) the cost of an elaborate storage oscilloscope or data capture system.

Lacking any precise description, many observers dub them personal instruments or computer instruments and a recent report in the top professional journal, *Electronics*, gave added respectability to this interesting new development.

Cheaper by a wide margin than conventional test instruments, which in many instances rely on built-in microprocessors, they make use of the mass produced computing power and memory of personal computers. They can thereby benefit from the economies of scale of units which often sell virtually in their millions, but without necessarily sacrificing any of the technical characteristics of instruments which may cost five times as much, and sell in mere thousands.

At present add on modules have been produced to do the job of Digital Storage Scopes, Signal Generators and Data Acquisition Systems, but new ideas are coming up fast and the range of applications is rapidly being expanded.

At present this budding new industry exists solely on the west coast of the United States, and although the computers which they favour, Apple and Commodore, are widely available over here, no British company, as far as we know, has caught on to this exciting new idea. Perhaps this article will stimulate somebody into action.

Northwest Instruments Inc. of Beaverton, Oregon and RC Electronics Inc. of Santa Barbara, California have

tended to specialise in the plug-ons to convert an Apple Computer into a sophisticated, dual-channel, storage oscilloscope.

In this application, operating software is supplied on floppy disc whilst the computer's display and keyboard serve as the scope's screen and control panel. Waveforms are processed by the computer and stored on disc. Data is acquired through probes or cables and results are displayed in a high quality scope format.

In addition to the above basic function relatively simple modifications enable the powerful processing capability of the Apple II to perform signal averaging and even digital voltmeter readout. A further development by the same firm is their "Instrumenware" software for applications like calibrating floppy disc drives and characterising amplitude and frequency in component testing.

Prices are not available for the UK, but in the States the plug on boards and the software to make them work, sell for about £350.

On a slightly different theme to Northwest Instruments, RC Electronics Inc. have tended towards the capture of transient phenomena with their variation of an Apple, storage scope. Again the prices quoted are ridiculously low against what an instrument manufacturer would quote, but this time there appears to be a limitation in the restricted resolution of the 8-bit analog-digital converter, although the company now say that a 12 bit single channel plug-on is on the way.

Oscilloscopes or variations of them seem to be the most popular application at the present, and the other companies mentioned above all have their own scope adaptor modules.

Robots In The Home

It's been talked about amongst the computer buffs (or freaks if you like) for years. Hobbyist magazines have projects to 'build your own' and no respectable computer exhibition is without its little android wandering about saying,

"Hello". There are even competitions staged to pit robot against robot in purpose built arenas reminiscent of the old Roman Gladiators. Although in our civilized era it is just to see which one can escape through a maze the fastest.

Now the big money is moving in, and it would seem at last that the dawn of the domesticated robot is really about to begin.

In the United States (where else) the first commercial automatons designed specifically for home and educational use are beginning to emerge, causing some industry watchers to draw parallels with the early days of personal computers in the mid 1970's.

At Future Computing Inc. a market research firm based in Texas, Portia Isaacson believes that home robots are now at the turning point where personal computers were when Apple arrived on the scene in 1976. According to her estimates 1983 US personal robot sales will be a modest 2,000 units, but jump to around 8,000 next year and then shooting to 100,000 units annually by 1987.

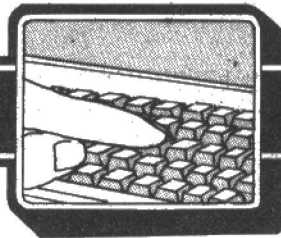
As with personal computers, personal robots will find applications in business as well as in the home, say the industry watchers. Both types of equipment will be sold through the same distribution channels as personal computers. By 1990, and that's only 7 years away, Future Computing reckons that personal robot sales will reach a staggering 1 million units annually in the USA alone.

Quite naturally there is keen competition developing to hit the jackpot with a home robot that will ride up on a wave of public interest just as did Apple five years ago.

The latest, and perhaps most startling to emerge is BOB (brains on board) produced by Androbot Inc. of Sunnyvale, California. BOB has eyes, ears and vocal chords and can be trained to navigate round rooms and corridors. However, the most significant factor about BOB is what and who is behind his creation.

Androbot Inc. has been heavily backed by Nolan K. Bushnell, who sired what turned out to become a multi-billion dollar, video games business called Atari. There is every intention to try and repeat this exercise with personal robots, and BOB could just get him a repeat performance.

BOB the clever robot from Androbot Inc.



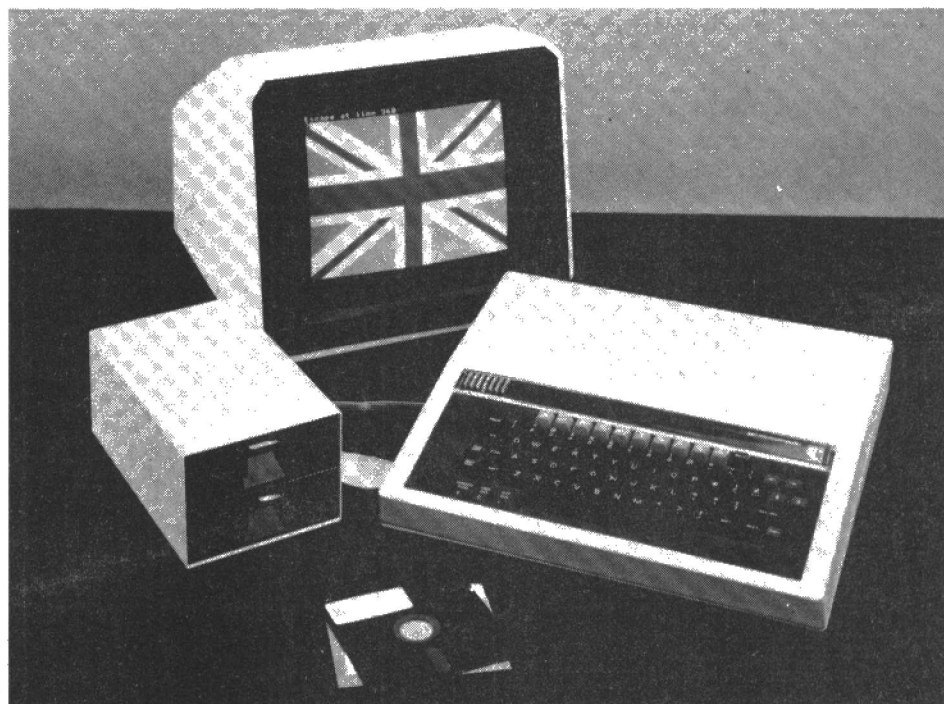
Dragon Data Approval For Salamander Software

Salamander Software have received full Dragon Data approval for their range of Dragon 32 software.

The range of games presently available for the Dragon is as follows:- Dragon Trek; Wizard War; Vulcan Noughts & Crosses; Games Compendium D1; Golf; Grand Prix.

All future releases of Dragon Software from Salamander Software will be cleared for approval with Dragon Data prior to release.

For further information contact: 17 Norfolk Road, Brighton, East Sussex BN1 4AA. Tel: 0273 771942.



FORTH For The BBC Microcomputer

JWB-FORTH V2 is a FORTH language ROM, which simply plugs into a spare location in the BBC machine.

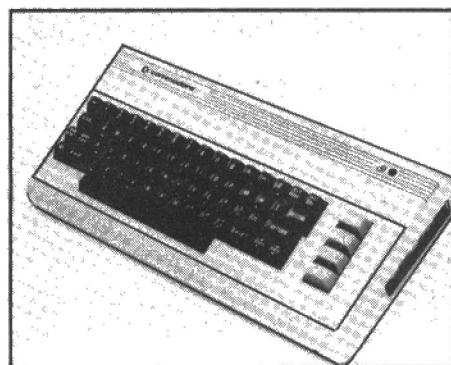
It will work with ANY current operating system, i.e. 0.1, 1.0 or 1.2 and is fully compatible with DISC and NET use, as well as cassette.

It allows FULL use of ALL graphics modes - No cassette version does this. ALL MOS calls are supplied. A Text and String editor are included.

JWB-FORTH V2 is the first extra language available on the BBC computer and is now being used by educational establishments, hospitals, The Ministry of Defence and the BBC as well as by many individuals. The program is fully supported and extensions include full floating-point representations and Graphics extension.

The program is available ex-stock and a manual for beginners 'WELCOME FORTH' is also available ex-stock.

For further information contact: HCCS Associates, 533 Durham Road, Low Fell, Gateshead, Tyne and Wear NE9 5EY. Tel: 0632 821924.



Software for Commodore 64

Audiogenic Ltd. - the Reading-based software manufacturing and distribution company have released a comprehensive range of software for the Commodore 64. Marketed under the 'SOFTWARE 64' label - a product line unique to Audiogenic, the following products are available, either directly from Audiogenic or via the nationwide Commodore Dealer Network and major chain stores: Motor Mania; Renaissance; Grandmaster; FORTH; MONITOR; Woodcraft 64.

For further information, please contact: Audiogenic Ltd., PO Box 88, Reading, Berks.

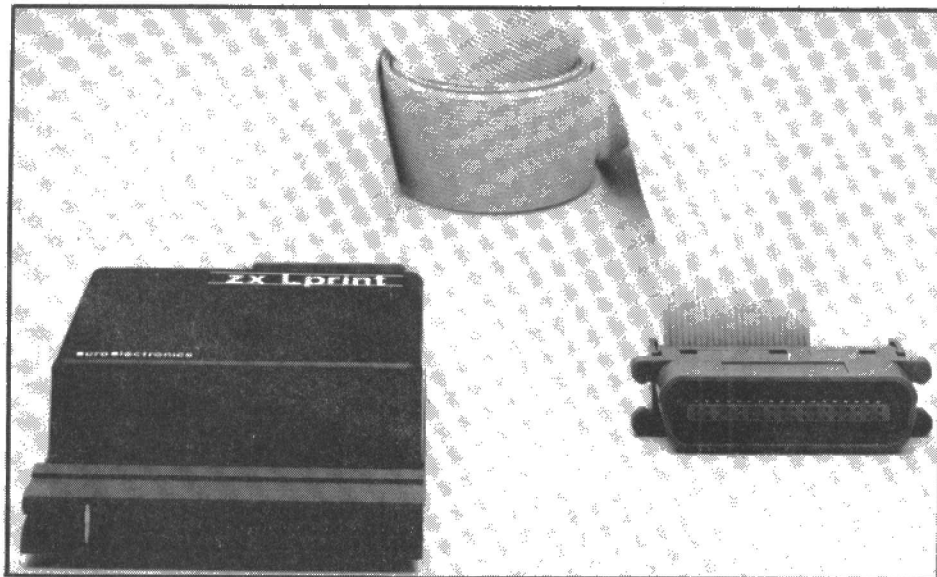
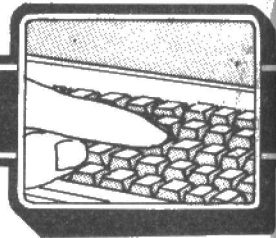
New 14" Monitor

The CE370 14". This advanced design cuts component count and eliminates the need for a heavy mains isolating transformer, reducing the weight of the CE370 to around 12 kilos. The CE370 is enclosed in a moulded case, with a moulded in handle giving portability without losing any design lines. This monitor has been designed with rough handling and safety in schools utmost in our minds.

The circuit board can drive a 14", 16" or 20" tube, and with the Phillips AX tube technology, will need no convergence or E-W adjustment. The 90° AX tube gives perfect colour registration with reliable and stable operation. Other features include Automatic drift compensation which automatically adjusts for tube ageing. Automatic signal selection accepts positive, negative separate or composite sync.

Prices start: CE370 A, R.G.B. £199.50. CE370 B, R.G.B./Composite Video (PAL)/Audio £250.00.

For further information contact: Cabel Electronic, Mount Road, Burntwood, Walsall, England WS7 0AX. Tel: 021 308 7075. Telex: 339671 ALD FAB.



Parallel Interface for Spectrum

Euroelectronics of Cheltenham have developed a parallel Centronics interface for the Spectrum ZX LPRINT. It plugs directly into the Spectrum rear connector and, via a ribbon cable, into almost any dot matrix or daisy wheel printer.

Any number of characters per line up to the maximum allowed by the printer in use can be printed with LPRINT command. LLIST gives the programme listing complete with Sinclair tokens (e.g. SCREEN\$, RANDOMIZE etc.) Printers which require special control characters from the range of 128 to 255 to access some of their functions can be used and controlled with ZX LPRINT interface. Sinclair tokens which fall into this range can be switched off using LPRINT CHR\$ 5 command and instead a true ASCII character can be sent to the printer. Return to Sinclair character set is via LPRINT CHR\$ 4. This feature enables the user to get the full use of his printer's capabilities. E.g. block graphics can be printed on some dot matrix printers or a form feed and eject can be implemented on letter quality printers. But this is not all...

A COPY command will dump the complete screen to a high resolution graphics printer. This opens a whole new range of applications. COPY software to go with ZX LPRINT interface is supplied on a separate cassette. LPRINT and LLIST commands do not need any extra software – they require just ZX LPRINT interface plugged into the computer.

For further information contact: Euroelectronics, Zlin House, Oakfield Street, Cheltenham, Glos. GL50 2UJ. Tel: (0242) 582009.

NANOS "quick-reference" card for the SINCLAIR ZX80/ZX81

A new NANOS "quick-reference" card has been recently prepared and published for the SINCLAIR ZX80 and ZX81 home computers. This card is in a fold-up accordion-style with 10 panels/20 pages, and it contains all the basic

information necessary for programming the ZX80 and ZX81 computer.

This card has been written in the U.S.A. by Paul Nanos of Nanos Systems Corp. As the author says, "Why waste your time and frustration struggling with the books? This card has it all".

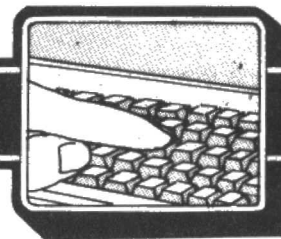
The NANOS cards are available for many other popular computers, e.g. Apple, Tandy, and they are being exclusively imported and distributed in the U.K. and Europe by ELKAN ELECTRONICS.

Contact Barry Elkan, Elkan Electronics, 11 Bury New Road, Prestwich, Manchester M25 6LZ. Tel: 061 798 7613 (24 hour service).

Eprom Programmer For The BBC Computer With Auto Run Facility

The ATPL EPROM Programmer has been developed for use with the BBC Micro to meet the demands of Educational users. Software development houses and dedicated hobbyists.

The board is a universal eprom programmer which has the facility for auto-running of programs situated in on-board eprom. It is possible to program, verify, read and check for blank, the following single rail eproms: 2516, 2716, 2532, 2732, 2564, 2764, 27128, 27256. Data files or programs (BASIC or machine code) may also be loaded or dumped from/to cassette. Programming etc. is done in a Low Insertion Force socket, which is automatically configured for the eprom being used by 3 relays under software control. The programming voltage is set for 25 volts, but may be changed to 21 volts by fitting a shorting link on the board. 21 volts is required for some versions of the eproms. These voltages are derived from an on-board switching regulator. The auto-run facility is available for the 27-



series eproms only. There are two 28 pin eprom sockets to hold the user programs and the Boot routines are contained in a 2732 in a separate socket. These routines are all less than 256 bytes long, so there is room for 16 different Boot routines, in 1 x 4K byte eprom. These are selectable from a 4 way DIP switch next to the socket. We are providing routines to run BASIC and machine code programs on O.S. versions 0.1, 1.0, 1.1, and 1.2 and also the facility for the Auto-Run to be by-passed.

The board is enclosed in a metal box with no 'bottom' to allow easy access to the auto-run sockets.

For further information contact: *Electro & Graphic Products, 37 Darton Road, Cawthorne, Nr. Barnsley, South Yorks. Tel: Barnsley (0226) 790609.*

Retraining Course in Electronics, and Computing

The course is broad based and consists of ten lecture modules on: Solid State Electronics, Computer Hardware, Computer Software, Microprocessors, Computer Aided Design, Signal Processing, Communications, Power Electronics, Control and Industrial Information Systems. All students carry out a project in electronics and computing during the course.

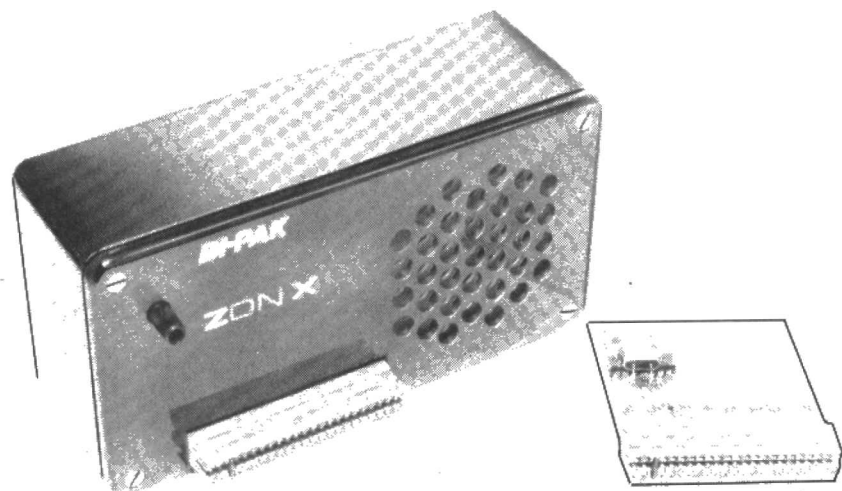
The course has been running successfully for four years. Students may attend full-time for one year, and industrially based students may attend part-time one full day per week for two years. The course may be suitable for those who wish to **convert** from some other discipline or for those who wish to receive up-to-date **re-training** in Electronics, Computing and Information Technology.

Successful applicants are required to have a good honours degree in Electrical or Electronic Engineering or Physics or an equivalent qualification such as corporate membership of the IEE. The course is recognised by the Science and Engineering Research Council (SERC). Companies sponsoring students may be eligible for a grant from the Engineering Industry Training Board (Course Code S413). The Manpower Services Commission supports a number of awards to mature applicants.

The course also provides suitable training for students who wish to pursue research for a Ph.D. degree.

For further information contact *Dr. M. S. Raven, Course Organiser, Department of Electrical and Electronic Engineering, The University of Nottingham, University Park, Nottingham NG7 2RD. Tel: (0602) 56101 ext. 2190.*

London Electronics College have now received official approval for their Computer Training Course. Prospectus can be obtained from: **London Electronics College, 20 Penywern Road, Earls Court, London.**



PLUG IN ADAPTOR FOR SINCLAIR SPECTRUM

Sound With Sinclair

Just three months ago Bi-Pak Semiconductors published details of their first sound generator for use with the Sinclair ZX81 computer thus adding much needed sound to an otherwise silent computer.

So successful has this proved with Sinclair users that Bi-Pak have now introduced a new modified version for use with all Sinclair computers i.e., ZX81, Sinclair Timex 1000 and the Sinclair Spectrum.

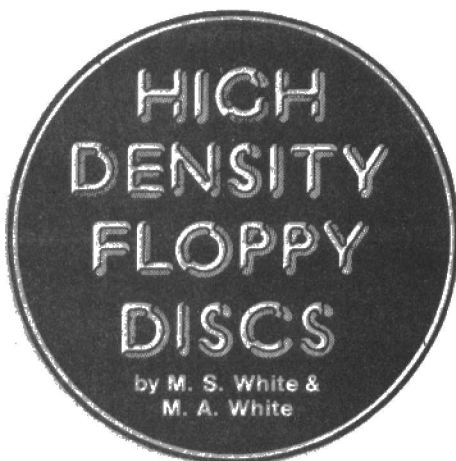
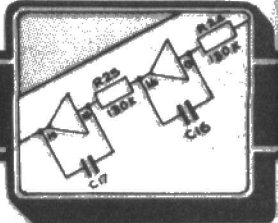
Designed ZON X the unit is self-contained in a black plastic case with a loudspeaker and manual volume control. No power supply or batteries are required to power the unit - it simply plugs into the rear of the Sinclair Computer.

The new unit offers a wide range of sound effects. These are obtained using the three-channel-plus-noise sound chip and is designed so that the pitches and volumes of the three channels are overall attach/decay envelope can be controlled by simple basic statements. This means that Pianos, Organs, Bells, Helicopters, Lasers, Explosions etc., can be simulated and added to existing programmes.

For use with the Sinclair Spectrum Computer there is a further plug-in adaptor for the ZON X which houses a crystal and other electronic devices needed to give **unlimited** sound facilities.

Bi-Pak Semiconductors have also established several U.K. Distributors but many areas remain open at this time, together with overseas appointed agents. Countries already covered include: Australia, Denmark, France, Germany, Holland and Switzerland where initial sales show great potential.

Trade and Overseas enquiries are welcomed. Contact R. Baines at Bi-Pak Semiconductors, The Maltings, 63a High Street, Ware, Herts. SG12 9AG.



The Floppy Disc Market

Memories are big business. There may be a world wide recession generally but the growth rate in the microcomputer industry is phenomenal. A 1982 estimate of the world market for hard and floppy disc drives is 12 billion dollars rising to 19 billion dollars by 1984. A recent IRD report predicts that sales of floppy discs will have passed a billion per year by 1991 with Winchester discs at 100 million and a mere 1 million for optical discs.

With this rate of growth the memory industry may be likened to something resembling a human ant hill. The level of work at present being undertaken is fast and ever changing. New materials and new methods are being researched into in an effort to win, or at least stay in the race to supply the growing market with ever smaller and denser storage capacity.

Early Development of the Floppy Disc

The story begins around 1967 at IBM where service engineers designed a new kind of disc for diagnostic purposes. By 1971 production of a new form of magnetic storage had begun. Within a few short years these discs were being used world wide for they were ideally suitable for microcomputers. The original disc was 8" in diameter and was very thin and completely flexible hence it became known as the 'floppy disc' (For the background to the conventional floppy disc see Trevor George in *Electronics and Computing Monthly*, August 1981, page 21). This first floppy was a 'read only' unit and had a capacity of 81.6 Kbytes. Two years later, in 1973,

its successor had a capacity of 243 Kbytes, an achievement made possible by increasing the number of tracks from 32 to 77 and from 1594 b.p.i. to 3268 b.p.i.

This was the model that found a large market and has led to the present high level of consumer demand. Both of the above discs recorded on one side only and the next model, emerging in 1976, was a disc recorded on both sides thereby doubling capacity to 568 Kbytes. One year later capacity again doubled and in 1979 with the introduction of the 72MD in which the disc speed increased to 720 rpm the data rate again doubled.

It may be seen, therefore, that from 1967-1980 work carried out by IBM resulted in the evolution of the flexible magnetic disc from one with a diameter of 8" with a capacity of 81.6 Kbytes and read only to the present one with a capacity of 1.2 Mbytes. In one decade IBM have increased the read/write capacity of their discs fifteen times, the data rate has increased by a factor of thirty and the average seek time has been reduced by a factor of eight.

Even higher capacity floppy discs are possible by recent developments. The biggest breakthrough in the past year is that of vertical recording. This technology which is based on the sputtering of thin magnetic films, promises an area density of four to ten times more than that which longitudinal recording can offer. In an industry where small is not only beautiful but means much more business, this is where the race is now at its most fervent. To proceed further some idea about this new technology is necessary.

Sputtering and Plasma Processing for Vertical Recording

In The electronics industry sputtering refers to the method by which surface atoms are ejected from a material by atomic or ionic bombardment. This may be achieved by creating ions in a glow discharge and accelerating these ions towards the material to be sputtered. The sputtering kinetics can be enhanced by introducing complex reactive gases into the discharge or plasma. The process is then referred to as plasma etching and plasma deposition. Sputter-

ing and Plasma processing are widely used in the electronics industry particularly for depositing and etching thin metal films and thin insulating films in device fabrication. More recently, these processes have been used to produce magnetic films for high bit density floppy disc manufacture.

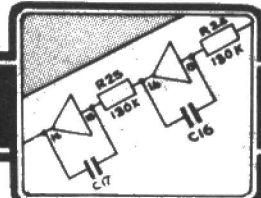
The Sputtering Process

The large number of ions required for sputtering may be obtained by applying an increasing potential across a gas until a glow discharge is created. Positive ions created in the discharge are accelerated towards the negative electrode and electrons accelerated towards the positive electrode. If the ions are sufficiently energetic and heavy they eject material from the cathode (known as the target) by sputtering, Figure (1). The anode is heated by electron bombardment. In the steady state this is the D.C. sputtering process and is suitable for depositing and etching conducting materials. If the material to be sputtered is an insulator a positive charge accumulates on the target and sputtering ceases. This may be overcome by bombarding the target with electrons or by using radio frequency sputtering.

R.F. Sputter Deposition

In the R.F. diode sputtering technique the substrate (plastic disc, silicon wafer etc.) is mounted on one plate of a parallel plate diode arrangement. The material to be sputtered is mounted on the opposite plate. The space between the plates is first evacuated to about 10^{-6} torr and then filled with either an inert gas such as argon or an inert gas which contains some reactive molecules such as oxygen. A high frequency voltage is then applied across the diode and increased until the glow discharge appears between the diode plates. High rates of sputtering are obtained for frequencies in the MHz range. A particular 'legal' frequency of 13.56 MHz has been adopted to minimize radio interference.

The R.F. plasma has a low resistance and is close to zero or ground potential. If the R.F. potentials on both plates are equal and opposite then **both** target and substrate are bombarded by positive ions and electrons on alternate half cycles. However, because the electron



velocity (or mobility) is much greater than ion velocity a negative charge accumulates on both plates. This leads to the formation of a negative d.c. potential on both electrodes relative to the plasma. A space charge is formed between the plates and the plasma and positive ions bombard both plates sputtering material, Figure 2(a). The negative d.c. plate potentials depend on the capacitance and plate area ($V_{d.c.} = Q/C$). If the plate capacitances are identical then since Q is the same for the same R.F. potentials then the d.c. potentials are identical. Sputtering occurs equally from both target and substrate and no deposition occurs.

If $-V_{d.c.}(T) > -V_{d.c.}(S)$ then the target will sputter faster than the substrate and a layer of the target material will condense on the substrate. This may be achieved by

- R.F. grounding the substrate so that $-V_{d.c.}(T) > 0V$ See Figure 2(b)
- Making the substrate plate area greater than the target area so that $-V_{d.c.}(T) > -V_{d.c.}(S)$
- Applying an external d.c. bias such that $-V_{d.c.}(T) > -V_{d.c.}(S)$

In practice, methods (i) and (ii) are both used. Method (iii) is used to provide a variable d.c. bias.

Crystal Orientation and Vertical Recording

When certain materials condense from the vapour phase the crystallographic axis of the grains may all be aligned in the same direction i.e. there is preferred orientation. This has been observed notably for materials with close packed structures such as hexagonal cobalt alloys, face centred cubic crystals and crystals with zincblende and wurtzite structures. For the hexagonal crystals the preferred orientation occurs with the c-axis of the unit cell perpendicular to the plane of the substrate (Figure 3). This orientation effect is being exploited in the sputter deposition of Co-Cr magnetic thin film alloys where it is found that vertical magnetisation may be obtained. It is found that magnetisation corresponds to a single grain which may be less than 0.1 micrometre in diameter (3.94 microinches). If each grain could be detected as a digital bit this would

correspond to 2.54×10^5 bits per inch or 254KBPI. Such bit densities have been demonstrated in the laboratory. However, grain sizes of less than 0.01 micrometre are easily produced in sputter deposition so that bit densities of 2.54MBPI are possible. Assuming about 1000 tracks per inch this would give area densities of 2,540 mega bits per square inch. A phenomenal storage capacity for the home computer.

Present and Future Developments

One of the biggest problems with the floppy disc market, particularly where the home computer user is concerned, is the lack of compatibility between different makes of disc. One move to achieve compatibility is that recently announced by Sony. They, together with twelve other manufacturers Xidex, 3M, BASF, Atari, Athana, Fuji Photo Film, Shugart, Memorex, Verbatim, TDX, Media Systems Technology and Wabach Datatech, have agreed to support a compatible 3.5 inch floppy disc format.

According to Sony the major technological issues relating to compatibility have been settled. Compatibility will

strengthen the position of the 3.5 inch disc with a hard covering as the leading format for a microfloppy industry.

The recommended specifications for the 3.5 inch microfloppy are 80 tracks with a density of 135 tracks per inch. The maximum bit density will be 8187. The recording system will be MFM with

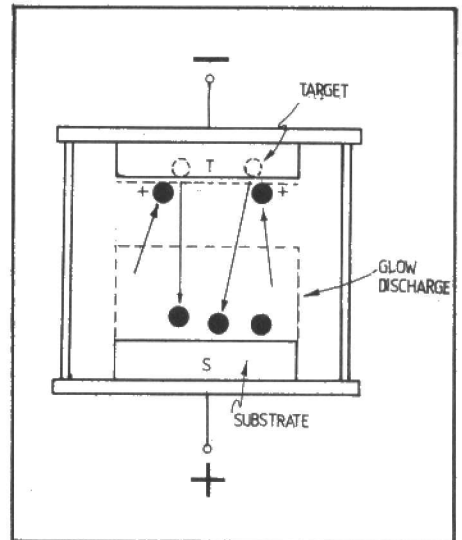
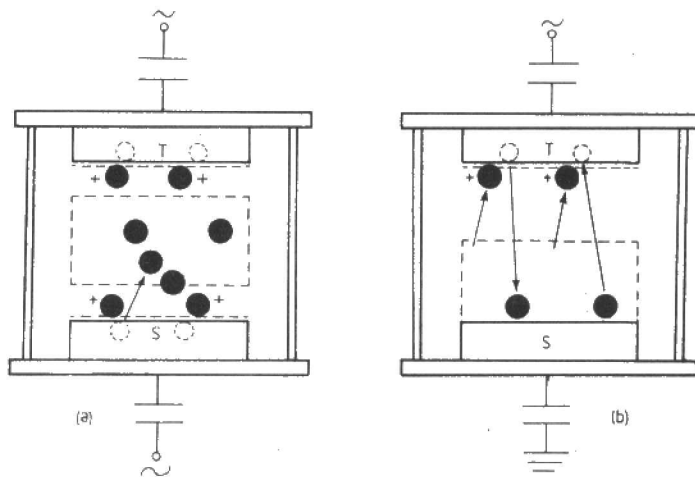


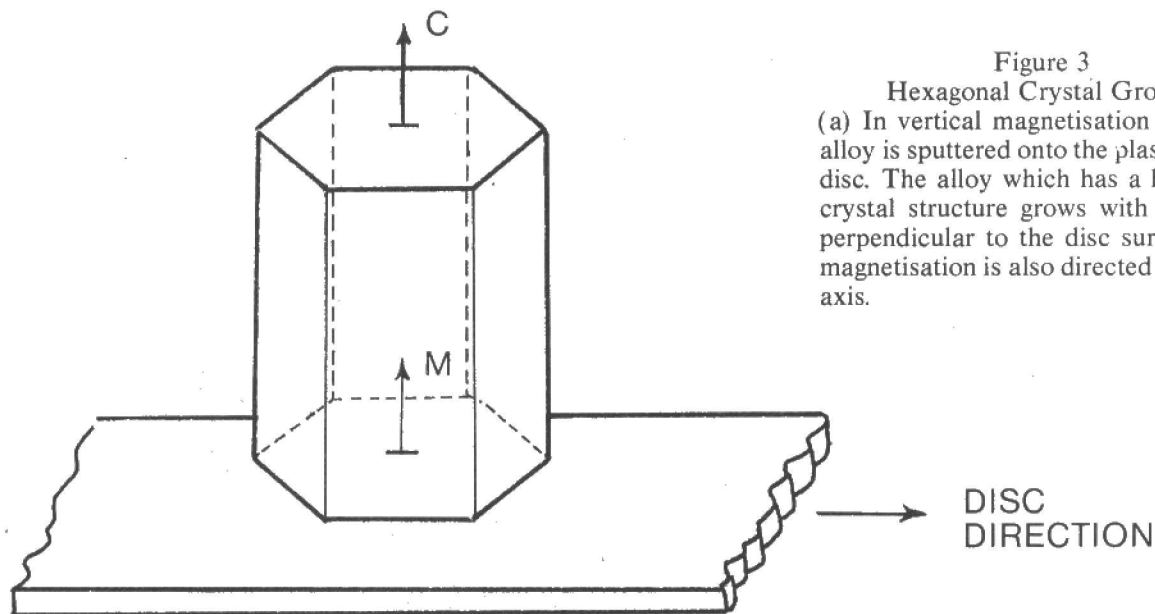
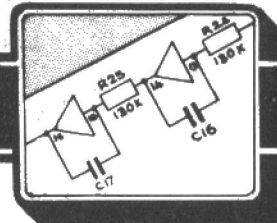
Figure 1

D.C. Sputtering system. Positive ions impinge on the negative target. Atoms are sputtered from the target and condense on the substrate (bottom).

Figure 2
R.F. Sputtering system



- (a) Equal substrate and target sputtering occurs if the electrode geometries are symmetrical about earth potential. (b) The target only is sputtered if the substrate is R.F. grounded.

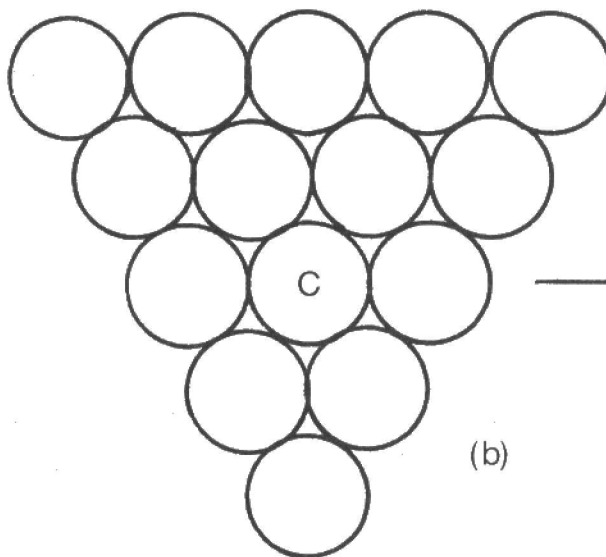


(a)

Figure 3
Hexagonal Crystal Growth
 (a) In vertical magnetisation of Co-Cr alloy is sputtered onto the plastic floppy disc. The alloy which has a hexagonal crystal structure grows with its c-axis perpendicular to the disc surface. The magnetisation is also directed along this axis.

a recording capacity of 0.5MBytes on each side. The diameter of the disc will be 86.00mm with the cartridge measuring 90 x 94mm x 34mm thickness. When other specifications have been determined Sony will be granting non-exclusive manufacturing licences to qualified media manufacturers. This should enable the standard to be widely adopted.

Meanwhile Toshiba are hoping to mass produce a 3.5 floppy disc with 3MBytes sides within the next two years. Toshiba claim they are the first company to develop a perpendicularly magnetised disc of this size. They have overcome the problem of manufacturing a magnetised surface by sputtering a 0.5 micrometre layer of chromium-cobalt alloy on both sides of a polyester base film. Toshiba claim that this represents a 27 fold improvement over existing floppy disc capacities. Although such firms as Vertimag may wish to argue with this claim there is no doubt that the Japanese consistent effort over many years in vertical recording research has given them an undoubted lead. One leading U.S. consultant estimates that the Japanese effort in vertical recording is at least twenty and perhaps as great as thirty times that of

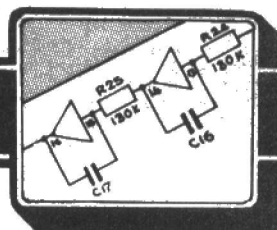


(b)

(b) This shows that the packing of equal cobalt atom spheres can lead to hexagonal symmetry with the c-axis perpendicular to surface.

their own. Moreover, this effort is not confined simply to commercial enterprises for at least twelve Japanese universities are active in the field. The figure for the U.S. is one. Whether this

seven year lead can be overcome by the belated recognition that has now forced leading U.S. and some European companies to take vertical recording seriously remains to be seen.



UNDERSTANDING DIGITAL ELECTRONICS

by J. Oliver Linton

Article 3 Sequential Logic

So far, all the circuits we have considered come under the heading of combinational logic because the outputs of the circuits are always completely determined by the present combination of inputs. The simple circuit shown in fig. (1) is fundamentally different and introduces a whole new world called sequential logic. In circuits of this type, the present state of the outputs depends not only on the current state of the inputs, but also on their previous state or sequence of states. In short, these circuits possess a memory.

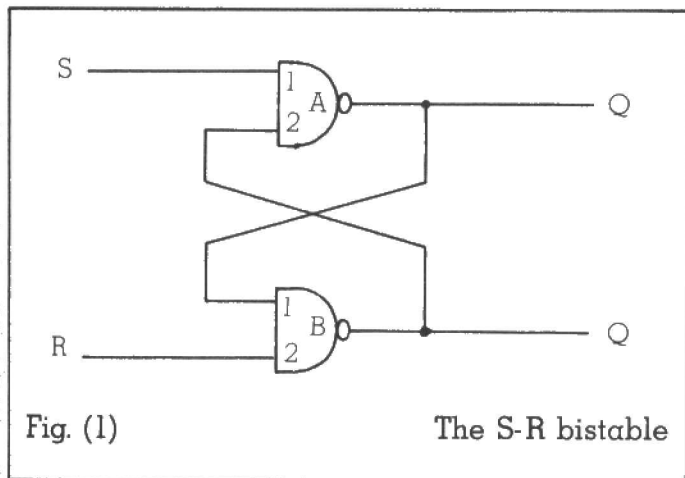


Fig. (1)

The S-R bistable

Referring to fig. (1), S and R stand for SET and RESET and are active LOW – that is to say they are normally held HIGH and taken LOW when required. Q is the output and as we shall see, Q is always the inverse of \bar{Q} . Suppose that Q is LOW and that both S and R are in their normal HIGH state. Because the input B1 is LOW, \bar{Q} must be HIGH. This in turn means that both inputs of gate A are HIGH and Q must therefore be LOW. This reinforces our original assumption that Q was LOW. The circuit is therefore in a stable state. Suppose that S is now taken LOW. This forces Q to go HIGH and if you trace the logic levels through gate B you will find that \bar{Q} goes LOW thus switching the circuit into a new stable state which remains even when S is returned to its former HIGH state. The circuit can be reset to its original state by taking R momentarily LOW. Since it has two stable states, this circuit is called the S-R bistable or latch. It is well worth getting familiar with this and other circuits to be described here by building them on a breadboard using the indicators described last month.

When building sequential circuits on a breadboard, unpredictable results can often be obtained due to the problem of switch bounce. Whenever an electrical contact is made by flicking a switch or pushing a wire into your breadboard, the

chances are that the contact will be made and broken several times. Sequential circuits operate so quickly that they will act on every spurious pulse generated. In order to eliminate this problem an S-R bistable is often used. A suitable circuit is shown in fig. (2).

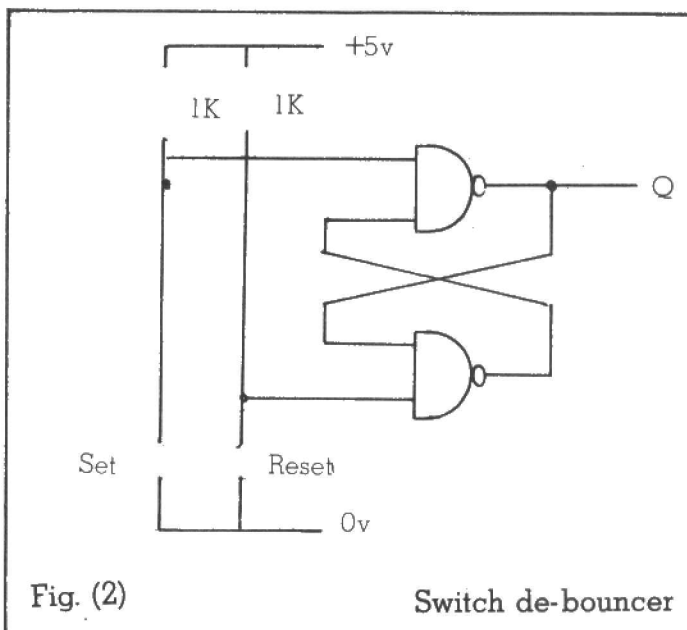
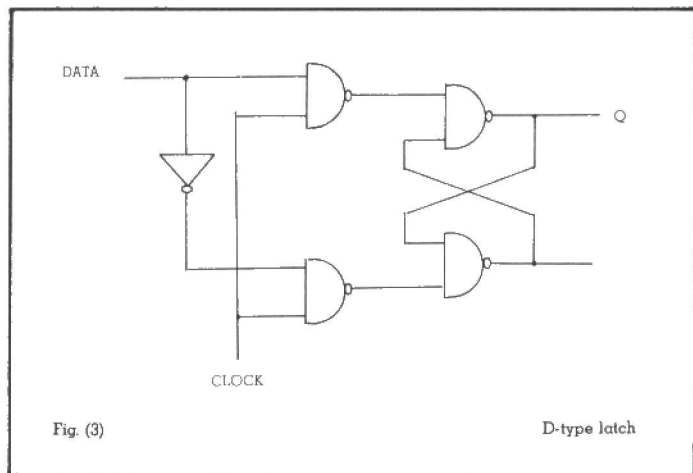
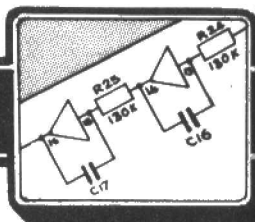


Fig. (2)

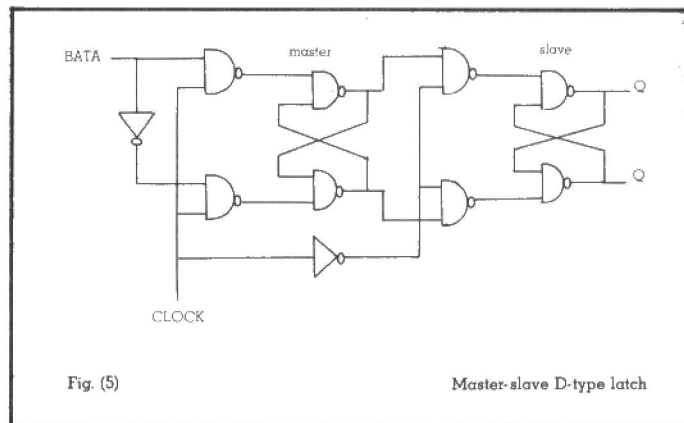
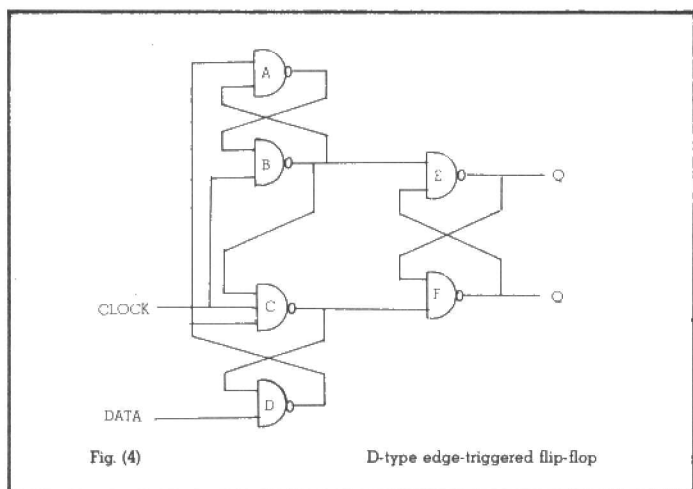
Switch de-bouncer

The snag with the S-R bistable is that if both S and R are taken LOW simultaneously, both Q and \bar{Q} go HIGH and the final state of the bistable when S and R again go HIGH will depend on which input stayed LOW for longest. It is desirable therefore to ensure that S and R do not go LOW simultaneously. This can be done by making R the inverse of S by placing an inverter between the two inputs. The trouble is that we lose the memory action of the circuit – because there is always one input that is LOW. We can restore the memory action of the circuit by adding a CLOCK input which enables the inputs only when taken HIGH. This arrangement is shown in fig. (3) and is called the D-type latch. As long as the CLOCK input stays LOW, Q and \bar{Q} will stay in their current states whatever the DATA input does; but when the CLOCK input goes HIGH, Q will assume the current state of DATA. Normally, steps will be taken to ensure that the DATA does not change state while the CLOCK is HIGH because the outputs will change state immediately. The latch is said to be 'transparent' because under these conditions the output can 'see' the input.

The 7475 is a dual 2-bit transparent latch. In fact it contains four D-type latches each with D, Q and \bar{Q} lines; but due to the shortage of pins, the CLOCK lines for the latches are coupled together in two pairs labelled usually E0-1 and E2-3.



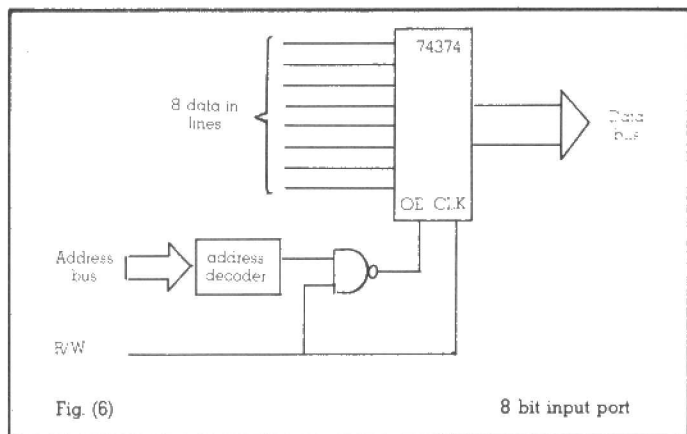
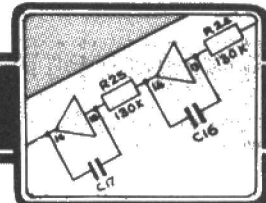
A more useful type of latch is known as the D-type edge-triggered flip-flop and its circuit is shown in fig. (4). In operation it acts just like the ordinary D-type latch except that as soon as the CLOCK goes HIGH the DATA input is 'locked-out' and further changes in the DATA do not affect the Q and \bar{Q} outputs. It works as follows. When the CLOCK is LOW, the outputs of gates B and C are forced HIGH and the outputs Q and \bar{Q} are unable to change. Changes in the DATA at this time will merely cause gates A and D to swap states. At the instant that the CLOCK goes HIGH, gates A/B and C/D latch onto the state determined by the DATA line causing the output of either B or C to go LOW thereby setting the output Q appropriately. At the same time, however, the DATA line is 'locked-out' against further changes because if the output of B is LOW, output C **must** be HIGH. Similarly, if output C is LOW, gate D is disabled and the DATA line ineffective. It is only when the CLOCK goes LOW again that both B and C go HIGH and the DATA line can again influence the states of the gates A and D. If you wish to make this circuit, you will have to redesign it slightly by replacing the three-input NAND gate with 2 two-input NAND gates and an inverter but it can still be done using just 2 7400's. Alternatively you can try some experiments with a 7474 which contains two D-type positive edge-triggered flip-flops. Either way, do not forget to use a debounce circuit on your CLOCK input otherwise some very peculiar things will happen.



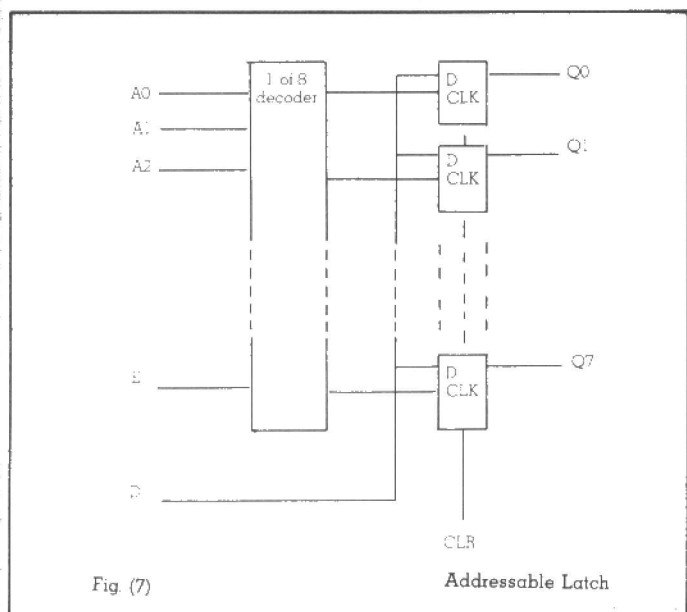
A second approach to the problem of locking-out the DATA when the CLOCK is HIGH is to use a master/slave system of latches as shown in fig. (5). The left hand side is a conventional D-type latch but is followed by a second latch whose CLOCK is inverted. When the CLOCK goes HIGH, data is read from the DATA line into the first latch but it is only transferred to the second latch at the instant the CLOCK goes LOW. It will be seen that the operation is negative edge-triggered rather than positive edge-triggered but an inverter in the CLOCK line could change that. Note carefully that although the first latch is enabled when the CLOCK goes HIGH, the data which is actually transferred to the output when the CLOCK goes LOW is the data on the DATA line **at that instant**. This is because the first latch is a transparent latch. If the master is replaced with a positive edge-triggered latch, the data is **read** on the **leading** edge of the CLOCK pulse and **written** out to the output lines on the **falling** edge of the pulse. In this case we say that the flip-flop is **pulse-triggered** rather than **edge-triggered**. Pulse triggered D-type latches are not very common but the very popular J-K flip-flop (to be described next month) is almost invariably pulse-triggered (the 74109 is an exception). Here is a selection of latches and flip-flops:

- 7475 Dual 2-bit transparent latch
- 7474 Dual D-type flip-flop (with Set and Reset lines)
- 74175 Quad edge-triggered D-type flip-flop (with Master Reset)
- 74174 Hex edge-triggered D-type flip-flop (with Master Reset)
- 7473 Dual positive pulse-triggered J-K flip-flop (with Reset)
- 7476 Dual positive pulse-triggered J-K flip-flop (with Set and Reset)
- 74109 Dual positive edge-triggered J-K flip-flop
- 74113 Dual negative edge-triggered J-K flip-flop

A group of latches or flip-flops with common CLOCK and RESET lines is properly called a register. In a microcomputer, registers are found not only in the machine's memory but also inside the microprocessor itself where they are used to store instructions and data, and wherever the microprocessor has to communicate with other 'peripheral' units like the keyboard, the VDU or the cassette recorder. Most microcomputers have a bi-directional data bus – that is to say, a parallel bundle of wires on which data can flow both out of the microprocessor to the peripheral units or vice-versa. Now it is clearly undesirable

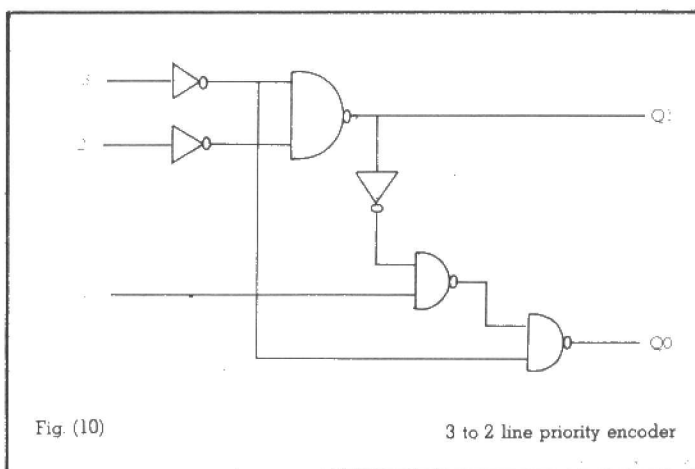
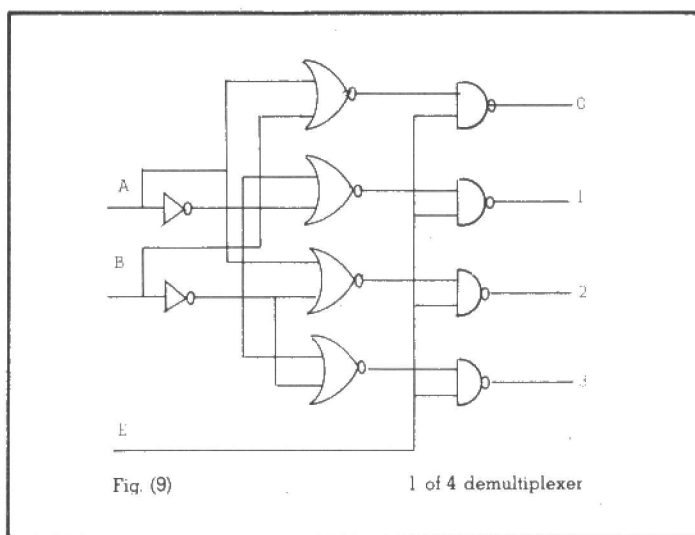
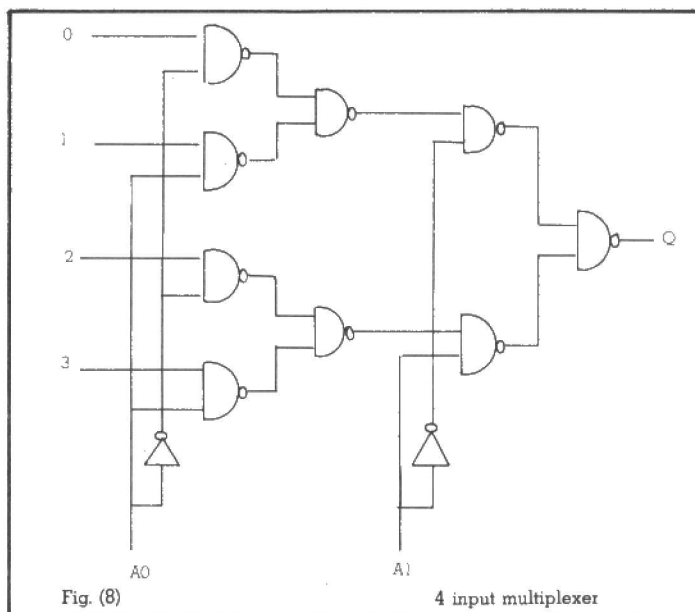


to have two such units trying to put different information onto the same bus at the same time. The registers which connect to the bus are invariably, therefore, of the three-state variety. What this means is that the outputs can either assert a **HIGH** state or a **LOW** one, or enter a third state which presents a high impedance to the bus thus allowing other registers to use it without interference. Obviously steps must be taken to ensure that only one register uses the bus at any one time (though of course, any number of units may **read** from the bus simultaneously). The 74374 is an octal D-type flip-flop with three-state outputs and the diagram in fig. (6) shows how it might be used to provide an 8-bit input port for a microcomputer with an 8-bit data bus. The outputs must only be enabled when a) the address of the port appears on the address lines, and b) when the R/W line is **HIGH** (i.e. the microprocessor is ready to **read** the data bus). The R/W line can also provide the clock pulse so that the most recent information is read.



While we are on the subject of three-state outputs, it is often necessary to provide a simple buffer stage between a device and a bus. The 74125 and 74126 are quad three-state buffers while the 74245 is a complete 8-bit bi-directional buffer with Enable and Send/Receive lines packed into a slim 20 pin package. You will find one of these, for example,

buffering the 1 MHz extension data bus on the BBC micro-computer and beside it a 74244 (the unidirectional version) buffering eight address lines.



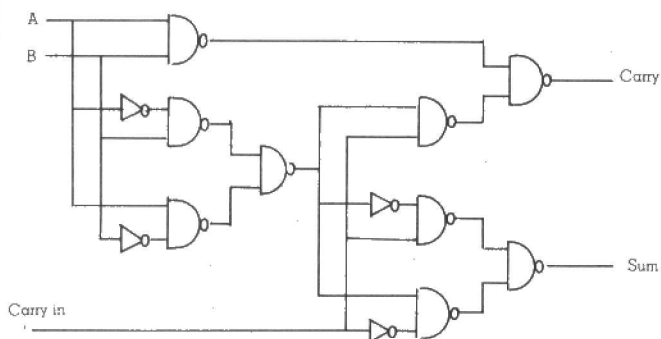
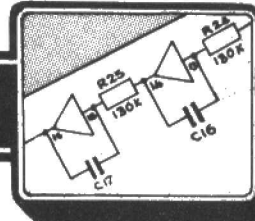


Fig. (11)

Full adder

To conclude this month we will take a quick look at a very simple memory unit—the addressable latch number 74259. Its circuit is shown in fig. (7) and is quite easy to understand. The three address lines A0, A1 and A2 are decoded into eight CLOCK inputs which select the appropriate latch. The DATA line is connected to all the latches, but since only one latch is addressed at any one time, the data is only entered into one latch. The chip is also provided with a CLR pin which resets all the latches to zero and an Enable pin which allows the addresses to be changed without setting or resetting the wrong latches. It is only a short step from here to understanding how a complete Random Access Memory chip works, but before we look at memories we shall explore some more aspects of sequential logic next month including counters and shift registers. In the meantime, figs. (8)–(11) give possible solutions to the problems suggested last month: if you didn't figure them out, why not try them on your breadboard?

MERLIN (Micro Systems) Ltd.

SOFTWARE ON TAPE FOR THE UK101 AND SUPERBOARD

Please describe screen size (UK101/Enhanced UK101/Enhanced Superboard) and Monitor (CEGMON, MON02 or Synmon/MON01).

- LE PASSE-TEMPS You NEED this one if you haven't already got it. (£3.00)
- GALACTIC HITCHHIKER An adventure, all in machine code. A beauty! (£7.00)
- SUPERTREK Sail boldly through the universe zapping moving Klingons in real time. Superb graphics (£7.00)
- KAISER OTHELLO This program is unquestionably the best Othello player we know. There are six levels but we have never yet won beyond level three! (£7.00)
- DRAGON'S LAIR A fantastic M/C Adventure program from the author of Kaiser Othello. (£6.00)
- LOST IN SPACE Another intriguing M/C Adventure program from the same source. (£6.00)
- STARTREK The old favourite, beautifully presented. (£6.00)
- LUNAR LANDER A real challenge. You won't get down in less than 3 hours. (£3.00)
- HANGMAN Excellent graphics, P.E. said so! (£3.00)
- BASIC TUTOR (8 x 4K programs) The only way to learn — at the keyboard. (£12.00)

SOFTWARE IN EPROM FOR UK101 AND SUPERBOARD

- MERLINS WORD (£17.50) Full feature word processor. Resident at S9000.
- NEW BASIC FOUR (£9.50) Includes named program SAVE/LOAD routines, "Old", Trace, Dynamic Halt.
- TOOLKIT (£12.50) Includes new SAVE/LOAD routines as BASIC 4, plus Trace, Dynamic Halt, Renumber, Single Step BASIC, Debounce etc. Resides at S8800.
- CEGMON (£15.00) Widely accepted now as the standard Monitor.
- EXTENDED MACHINE CODE MONITOR (£7.50) EXMON in EPROM at S8800 and it doesn't crash BASIC.

HARDWARE ADD-ONS FOR THE UK101 AND SUPERBOARD

- These kits are complete in every way, including switches, pre-formed cable assemblies etc. All you will need is some connecting wire and solder.
- MOTHERBOARD SYSTEM (£19.50) Now you can add on all those extras easily. Provides 8, yes eight, fully buffered J1 type sockets.
- BK STATIC RAM BOARD (£19.50) Excluding RAM.
- HI-SPEED CASSETTE INTERFACE (£19.50) At last a system that works. Completely RELIABLE 4000 baud (8000 with reasonable cassette) plus software on tape (see also EPROM Software) for named file handling. A delight to use.
- VIDEO ENHANCEMENT (£19.50) Switch selectable 16 x 48 or 32 x 48 display without butchering your computer. Sorry, but this one for UK101 only. Monitor EPROMS re-blown to suit for just £2.50.
- BK EPROM BOARD (£19.50) Takes 4 x 2K EPROMs located in any BK block.
- MONITOR BOARD (£9.50) Plugs into Monitor socket to provide switch selection of up to 4 EPROMs.

ECM04

Carriage/P.P. included. Please add 15% VAT to all prices.

93 HIGH STREET, ESTON, CLEVELAND. Telephone: (0642) 454883

HAPPY MEMORIES

Part Type	1 off	25-99	100 up
4116 200ns	.90	.81	.78
4116 250ns	.70	.63	.60
4816 100ns For BBC Comp	2.25	2.01	1.95
4164 200ns	3.99	3.56	3.42
2114 200ns Low power	1.15	1.00	.90
2114 450ns Low power	.95	.85	.80
4118 250ns	3.95	3.55	3.40
6116 150ns CMOS	3.55	3.20	2.95
2708 450ns	2.60	2.25	2.10
2716 450ns 5 volt	2.35	2.10	2.02
2716 450ns three rail	5.75	5.00	4.65
2732 450ns Intel type	3.50	3.15	3.00
2532 450ns Texas type	3.70	3.30	3.00

Z80A-CPU	£3.95	Z80A-PIO	£2.99	Z80A-CTC	£2.99
6522 PIA	£3.70	7002 A-D	£4.60	3691	£2.75
88LS120	£2.20	7805 reg	.50	7812 reg	.50

Low profile IC sockets: Pins	8	14	16	18	20	22	24	28	40
Pence	9	10	11	14	15	18	19	25	33

Soft-sectored floppy discs per 10 in plastic library case:

5 inch SSDD	£17.00	5 inch SSDD	£19.25	5 inch DSDD	£21.00
5 inch DSQD	£26.35				

8 inch SSDD	£19.25	8 inch SSDD	£23.65	8 inch DSDD	£25.50
-------------	--------	-------------	--------	-------------	--------

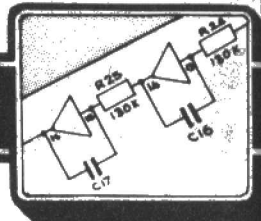
74LS series TTL, large stocks at low prices with DIY discounts starting at a mix of just 25 pieces. Write or phone for list.

Please add 50p post & packing to orders under £15 and VAT to total. Access & Visa welcome. 24 Hr service on (054 422) 618. Government & Educational orders welcome, £15 minimum. Trade accounts operated, phone or write for details.

HAPPY MEMORIES (ECM).

ECM05

Gladestry, Kingston, Herefordshire, HR5 3NY. Tel: (054422) 618 or 628



The "Soloload" Data Storage System

by R. W. Dawson

Most small computer systems, usually based on 8 bit microprocessors, use audio cassette tapes and recorders for "user program storage".

Since neither the cassette tapes or recorders are specifically designed for the storage of digital information, several difficulties are encountered with their usage in this application.

Amongst these difficulties are: the recording and replay times of programs are very slow, typically ranging from 2 to 10 minutes.

If more than one program is recorded on a single tape the replay time can be greatly increased unless a separate record of the approximate starting position on the tape is kept. In the worst case the program load time could be 1 hour.

Many of the programs used are purchased from software houses using high speed duplicating techniques, over which there appears to be no control of standards. The resultant recording quality and level therefore varies considerably, so that when a newly acquired program is first loaded a considerable amount of time can be spent adjusting volume and tone controls before a successful load is achieved. (The author had spent several hours on a newly acquired tape). For obvious reasons it is again necessary to keep a separate record of the settings required.

Programs stored on the audio cassette used with the computer system can prove impossible to use with another cassette recorder, since (especially with low cost unit - around £40) the head alignment within the recorder is not usually sufficiently accurate for digital usage. (This effect is often noticed even with audio reproduction).

Having established the apparently correct levels for the control settings on the recorder there are many occasions on which a previously satisfactory recording will fail to load first time and further adjustments have to be made.

Due to many effects including stray magnetic fields, heat, tape stretch etc., it is not unknown for programs stored on tape to become completely unusable.

Many users of small computer systems not only purchase pre-recorded programs but also write their own. If they use an "Editor-Assembler" so as to write in the more efficient machine code rather than the resident high level language (usually BASIC) it is possible to have to load three separate cassettes sequentially before program development can continue. Thus increasing the initial loading time by a factor of three, (say 10 to 30 minutes) assuming a successful load of each tape.

Tapes, cassettes and recorders are basically mechanical and electromagnetic devices and as such are inherently unreliable and require regular servicing and/or replacement if their already limited performance is to be maintained.

In an attempt to overcome the limitations of cassette based systems many manufacturers offer an (expensive) optional extra or upgrade device known as a "Floppy Disk Drive". The main advantage of these units being their faster record and replay times, however most of the objections that apply to cassette systems still apply to the Floppy Disk units.

A cassette recorder suitable for use with small computer systems will cost in the order of £30 to £100. Upgrading to a floppy disc, even if a suitable unit and interface electronics are available for the particular computer system will cost in the order of £400 to £800.

It was these problems which led "Solo Electronics Ltd." to investigate the possibilities of alternative mediums for program storage on small computer systems. To facilitate this a number of design objectives were specified. These are as follows.

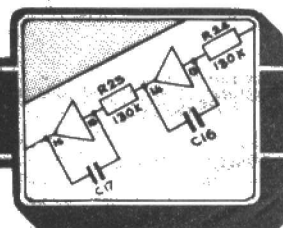
Design Objectives

1. To provide a means of storing and recalling user programs and data files at high speed with a high degree of reliability and data security using only solid state electronic techniques.
2. To provide storage mediums for permanent, semi-permanent and temporary storage of programs and data files.
3. The system shall be automated to the level that no specific knowledge of the computer or the storage system other than that already possessed by the user is required.
4. The unit shall use only the connections provided by the computer manufacturer for expansion/interfaces and therefore not require any modifications to the users equipment.
5. The program storage devices should be constructed in such a manner so as not to limit the maximum number of programs that may be used by the computer (i.e. similar to the cassette systems having no theoretical limit to its storage capacity).
6. The technique deployed should be useable on all computer systems.
7. The system shall be user friendly providing English prompts if interaction is required.
8. The operating system shall provide all the facilities necessary to enable existing tape based programs to be transferred to the new storage medium.
9. The storage system shall be compatible with any programming language used, in general this means BASIC or Machine Code programs.

Basic Principles

A. MEMORY TYPES AND THEIR USAGES.

The most commonly used permanent but re-usable semi-conductor memory device deployed in modern electronics is the "EPROM" (Erasable, Programmable, Read Only Memory).



There are several types available, the most common (and therefore lowest cost) being erasable by exposure to Ultra-Violet light, other types can be erased using electrical signals. These are known as "UVEPROMS" and "EAROMS" respectively.

For storage of temporary data it is normal to use "RAM" (Random Access Memory). The normally used "RAM" would be unsuitable, for data storage, in a system that must retain data after the power has been removed, since the stored data would also be lost, however the development of large capacity RAM's based on the "C.MOS" technology enables this difficulty to be overcome.

"C.MOS" devices consume minute amounts of power thus making them ideally suitable for battery back-up applications. Modern CMOS-RAMS consume only micro-watts of power in the standby mode, hence a very small (rechargeable) battery will enable data to be retained for extended time periods.

APPLICATIONS OF EPROMS AND CMOS-RAM

It is envisaged that EPROMS would be used to store programs and data that is required in a permanent basis, or at least would only be altered infrequently.

The main applications of CMOS-RAM would be:- 1. the storage of rapidly changing data files; 2. during the development of new programs.

The reasons for this being mainly one of speed. EPROMS take 50 milliseconds per byte to store information and the erase time is around 10 minutes. CMOS-RAM can be altered instantly but does require battery back-up.

B. CONNECTION OF SYSTEM TO HOST COMPUTER

Most small computer systems are designed around an 8 bit microprocessor. These devices have an inherent memory addressing capability of 64K bytes without resorting to special memory management techniques.

A typical small computer system only uses about half of the total "Memory Map", and usually only 16K of the total 64K is used for program storage. The other 16K being used by the resident program and for peripheral drive logic.

This allows, in the simplest form, approximately 32K of the memory map to be used for system expansion in the form of ROM and RAM devices.

The problems of the fully utilised "Memory Mapped System" does not preclude the usage of the proposed technique but does increase the complexity of the unit a little since any storage medium must be treated as a true peripheral rather than an extension of the memory map.

When programs are written to run in user RAM it is not normally possible for them to be relocated (by the user) into new memory areas and still be used.

Thus if programs are to be stored in the spare memory areas they must be automatically re-loaded into their correct execution area before they may be used. Thus the unit is not only capable of storing programs in the memory devices but automatically provides sufficient data as to enable the stored program to be re-located at its designated addresses and executed correctly.

The Management Software

The unit contains its own management software (stored in EPROMS) which enable the programs, and other useful functions to be accessed upon demand. The basic functions undertaken by the Management Software are:-

1. Automatically undertake all the actions necessary to transfer a program from User RAM into the chosen storage medium. The software allows for this to have been loaded either from the keyboard or from an existing cassette tape.
2. Provide an identification "Name" for each program stored.
3. Ensure that duplicate names are not requested.
4. Provide all the necessary information related to program location, size and execution to ensure successful reloading of the program. With BASIC programs this is entirely automatic.
5. Undertake calculations to enable more than one file to be saved in any Rompack.
6. Ensure before a program is saved that sufficient storage space will be available in the chosen Rompack.

7. Provide a directory of named files presently stored in each Rompack so that any file can be instantly loaded.

8. Undertake when requested a search and transfer of the required program into its correct execution locations.

9. Since solid state memory devices vary in size, typically from 256 bytes to 8K bytes it can be seen that a program cannot necessarily be stored in a single memory chip. It is normal practice in this situation to manually divide the program into PROM sized sections and then save each section in a separate chip. This process would not fit into the philosophy of the design objectives and hence the management software overcomes this difficulty and programs the correct number of chips automatically, without any user interaction.

10. The management software provides "Monitor" facilities that enable existing, even protected, programs to be transferred into this storage medium. It must be stressed that these facilities are provided to enable back-up copies only to be made, in fact the software will not allow a tape copy of a protected program to be made - thus protecting software suppliers against "Pirating".

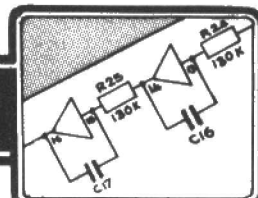
Future Developments

The initial models can only be used with 16K systems of the TRS-80 Model I Level II or the Video Genie Model EG 3003/4. An optional parallel printer interface board can be added into the Soloload's housing.

A model which is connected as a true peripheral, i.e. using the I/O Ports, is now in the final stages of development and should soon be available.

This more advanced model will be of particular interest to user's of fully expanded systems or those whom plan to expand their computer at a future date.

One of the major advantages of the new model is its increased storage capacity:- the present model is limited to 16K memory packs the new model has this capability expanded to 64K of memory. This is in fact an artificial limit dictated by economic reasons, it is quite possible to develop the system to enable hundreds of Megabytes to be stored and accessed without the need to change Rompacks.



Review of The BASICARE SYSTEM

by Stephen Adams

The idea for the Basicare system is split into two parts. One is the idea that the most important pieces of equipment are the peripherals, the "extras" that the microprocessor chip requires when building up a computer. These usually include RAM, ROM, INPUT and OUTPUT devices. If these can be made to work on a standard connection system then the only thing different between each computer would be the microprocessor's access to the rest of the computer system. In building the system Basicare have included all the necessary equipment to keep the system running without the help of the Processor. Things like refresh for dynamic RAM's has been included, hardware clocks for character sets and resets all take place INSIDE the system.

The second idea is that if the system connections can be standardised then the address lines can be extended to extend the memory map available. This extension consists more address lines which are set up by POKEing a memory address on the computer. Four data bits are then used with the other 16 bits of the standard address bus as a 20 bit address for any memory location. As 64K is the normal amount of memory that can be addressed by a 16 bit bus, adding another 4 bits (preset) extends it to 16 times 64K or 1 Megabyte (1 million addresses). Some of the memory however must be common to all PAGES (each of the 64K maps selected by the four bits is called a PAGE) as all require access to the ROM for instructions. If the ROM was switched out as well the microprocessor would lose control as it would have no instructions to carry on running the system. As Basicare have allocated 32K of RAM to a PAGE and has limited both the DATA and TOOLKIT areas (8K each) to the first four pages only, therefore the total amount of memory expansion including ROM's, character sets etc. is 576K. This is quite impressive, but remember that one 16K area must be allocated to the TV picture on the ZX81.

All of this expansion is controlled by a module called the MINI-MAP which provides the address switch and without it you are left with the normal 16K of address space.

The device which connects the microprocessor to the system is called the PERSONA and this allocates the areas of memory into segments. The segments are shown below for the ZX81.

Memory Map of the ZX81 using Basicare

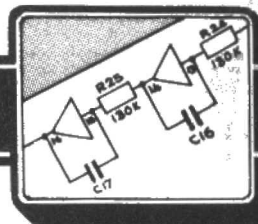
64K	FILE AREA B (SWITCH 2) 16K
56K	SLOT AREA (NOT SWITCHED) 4K
52K	PATH AREA (NOT SWITCHED) 4K
48K	DATA AREA (SWITCH 1) 8K
32K	FILE A AREA (SWITCH 2) 16K
16K	TOOLKIT AREA (SWITCH 1) 8K
8K	ROM AREA (NOT SWITCHED)
0K	

The two switches shown are independent of each other and switch 1 can only choose the first four pages.

The system is at the moment designed only for the ZX81 and the ZX Spectrum which have similar memory maps. No modules have been designed for interfacing any Sinclair bus devices to the BASICARE system, although the printer may be used if required by plugging it into the computer first. INPUT/OUTPUT port devices (like the Sinclair Printer) may be used. This is because Basicare monopolises only the memory mapped system to make it compatible to other computers like the VIC, ORIC etc. which have no such map.

File A

This is the main BASIC area on the ZX81 and the Spectrum. It is provided as standard on the Spectrum as a full 16K section where on the ZX81 only 1K is provided. The 16K RAM pack made by Sinclair would fit into this area. If using a PAGED system via the MINI-MAP module each PAGE must be initialised separately by selecting the page and then moving a peg at the back of the module to reset it. Once this is done programs can be written or LOAD'd from tape into each page by POKEing the PAGE number into the Mini-map and then inserting the program. Once this has been done and the programs started (either by Auto-Running them as in the manual or RUN from the keyboard) a jump from one program



to another may be done in BASIC or machine code to a program in a different PAGE. The variables stay with the PAGE that created them and the new PAGE starts off from where it was left last time. If you wish to carry variables between programs then they must be stored in the separate DATA section of memory (the method is up to you as no program is provided by BASICARE). The DATA section will not be switched by the jump to the new program.

If you don't use the mini-map then all you get is the normal arrangement of memory as on the ZX81 or Spectrum and the system is underused and expensive if no expansion is planned.

File A on the ZX81 also is reflected into FILE B so that the TV picture may be put out from FILE B, on the BASICARE system these files may be split, but at least one page must be in FILE B as the display PAGE.

CONSTRUCTION OF THE MODULE

Like all the Basicare modules the RAM's come in $6\frac{1}{2} \times 3\frac{3}{4} \times \frac{3}{4}$ inch grey plastic boxes. They are joined by a white 64 way edge connector at the top and pins on the bottom of the module. The PERSONA module is always on the bottom, but the rest of the Modules can be plugged into the top in any order. The case at the back has room for two holes, only the right hand side one being used on the RAM's. This contains a 18 way molex multipin connector which is used to select which 16K PAGE the RAM module sits in. Each set of four PAGEs is called a BANK and the BANK is selected by one blue clip and the PAGE within that Bank by another. PAGE 0, Bank 0, must be the first 16K of RAM to be allocated otherwise when you switch power up the system will crash. The rest of the RAM may be allocated to any PAGE. The third blue clip must be in the A/B position for PAGE 0, Bank 0 so that you have a TV display on switching on. If you wish to select A or B files for any other modules, then a wire jumper must be provided between the minimap module and the RAM module as there is not room to do this on the edge connector for the system. BASICARE can provide such a lead. Any 64K RAM module MUST sit entirely in one Bank covering all the Pages 0-3.

FILE B

FILE B is switched along with FILE B and usually contains the display file reflected from FILE A on the ZX81. If desired FILE A may be separated from FILE B as described in above. If you do so it will become just a data file for non-BASIC variables which can be PEEK'd or POKE'd. Machine code could be used to make the transfer from FILE B to FILE A where it will be used, a lot quicker.

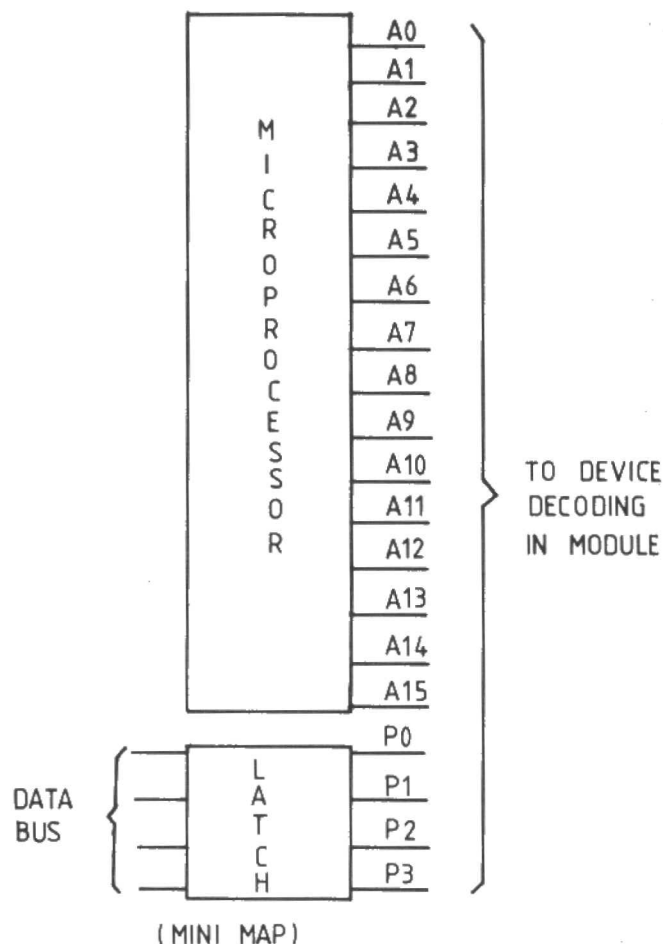
TOOLKIT Area

The Toolkit area is where machine code routines can be run on the ZX81 and is the only place that a user defined character set can be used. There are several modules that can be used in this area.

The DROM is a static RAM module that can provide up to 8K using 2K 6116 type chips. These chips are supplied by a 3.6 volt nickel cadmium, rechargeable battery when the power is turned off and can be used to store data or commonly used

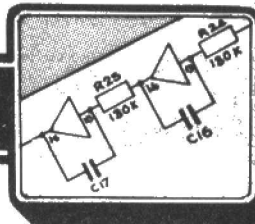
machine code routines. When the Power is ON the batteries will be recharging and the RAM's supplied by the computers power. This means that unlike ROM's they can be re-programmed, but still do not lose the data when the Power is switched off. A WRITE protect clip is also provided to stop accidental destruction of information. A USRFONT option is available to provide a user defined character set using the DROM to hold the characters. The module is supplied with one 2K chip, but extra chips can be supplied by Basicare or you can buy your own.

PAGE ADDRESSING



The RAM 08 Module

Again supplied only with 2K, but with a capacity for four chips inside the module, the RAM 08 can be placed in the DATA or TOOLKIT areas of memory. The USERFONT option can also be supplied, but it can only be used in the TOOLKIT area. If used without mini-map and the DATA area is selected it will also appear in the TOOLKIT area. The TOOLKIT area on it's own can be selected via blue connectors on the connection strip at the back of the module.



This is the area which is much overused at the moment on the ZX81 and BASICARE have eliminated this by not providing any Sinclair type bus interface and thus any competition for space. Modules may be provided for other manufacturers to interface their equipment at a later date. New modules forecast by BASICARE are discs and a colour board. One suggestion for the TOOLKIT area is for a machine code program to control all this extra memory and peripherals, to replace BASIC.

DATA Area

This as it's name suggests is used to transfer data from one page to another. As it can be switched independently from the RAM pages, the selection of data is up to the user. The TOOLKIT area is also on the same switch, so one goes with the other. The DROM module cannot be used in this area, which is a pity as common data cannot be switched between pages without using the TOOLKIT area, which might well be controlling the program. As with the TOOLKIT area this data area ignores the BANK switch and appears four times repeating the same page numbers in each Bank.

Variables would have to be transferred to and from the BASIC area by PEEKs and POKEs or a specially written machine code program. Using the minimap is the only useful way of making use of this area as no machine code routines can be run here on the ZX81.

PATH Area

This at the moment is unspecified, except for "further development". I presume this will act in some way similar to the BBC machines' TUBE interface with allows different processors to talk to each other.

SLOT Area

This is where all the physical devices sit, such as mini-map, printers, ports etc. It is decoded into seven slots and each device has it's own set of addresses within that slot. The device type within each slot is specified by Bits 6 & 7 of the address, up to four devices of the same device type may be used. The individual addresses for inside the device are allocated to the last five bits of the address thus giving at a maximum of 32 addresses per device. What this means in plain language is that the seven slots are each divided up into four different types. For instance there are three different PERCON modules, two for 24 Bit input/output ports based on the 8255 type device and one for a Centronics printer using the same IC.

The four device types can have a maximum of four of the same devices working at the same time. Each device having 32 unique addresses of it's own.

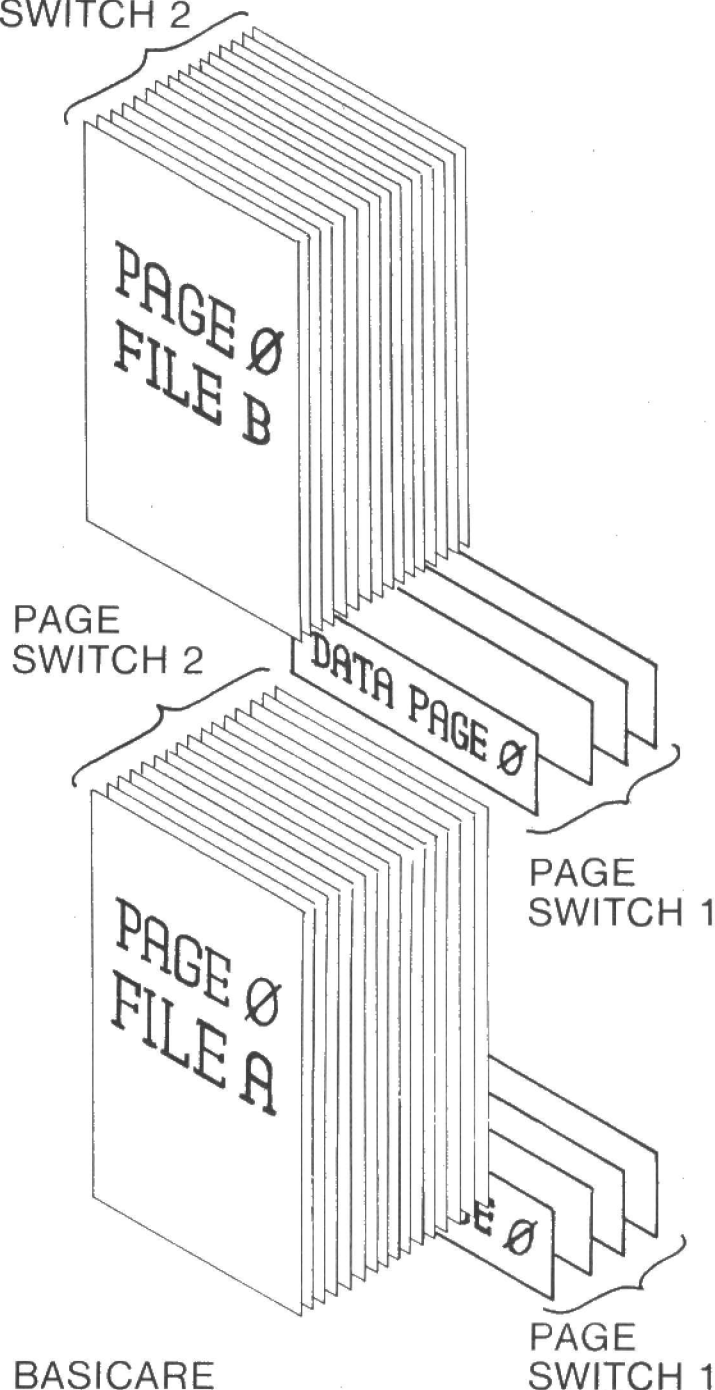
Only SLOT 6 (PERCON) and SLOT 0 (Mini-map) are specified at the moment. Other devices will be allocated slots when they arrive.

CONSTRUCTION OF A SYSTEM

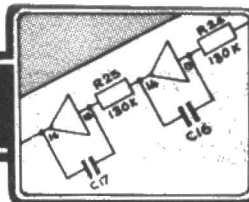
The idea of changing the computer modules is limited by the fact that software is needed to run it, so 6502 programs won't run on a system that uses a Z80A microprocessor. When considering a BASICARE system you must be after a system

with as much direct memory access as possible and with printers, disc etc. to build up to. To make the maximum use of this memory you must have available programs which will make the use of the system transparent to the user. At the present time you would have to write these programs yourself. Wordprocessing and data storage such as data bases are obvious examples of the type of use this system can be put to. A great deal of scientific computing could also be done on this system as it has more than enough input/output capability.

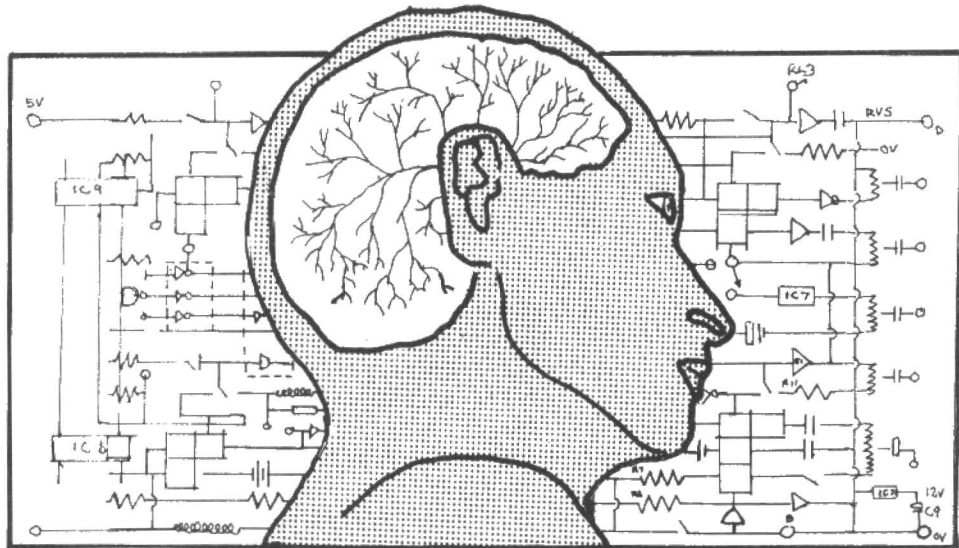
PAGE SWITCH 2



BASICARE PAGE SELECTION



THE COMPUTER BRAIN



on the description of shapes rather than shape recognition because this is not only more fun but more possible using standard microcomputer hardware.

Perimeters

The most obvious thing about any shape is the existence of a boundary that separates it from the background. Obviously if you can describe the boundary, or perimeter, of the shape then this is the equivalent of describing the shape (apart, that is, from saying what colour the region inside the perimeter is).

There are many ways of describing the perimeter of a shape. One of the oldest methods is to use an equation. For example, if you plot the graph of $Y=R^2-X^2$ you will see a circle. This is a powerful and very neat idea but if you try to extend the principle to describe any complicated or irregular shapes then you will be disappointed. There are far too many important shapes that cannot be easily described by an equation for this to be a universal method. The most common modification of the equation method to include a wider range of shapes is to use small sections of curves that can be easily described by equations to build up a more complicated one. For example, you can use a collection of short straight lines to approximate a curve. (See Fig.

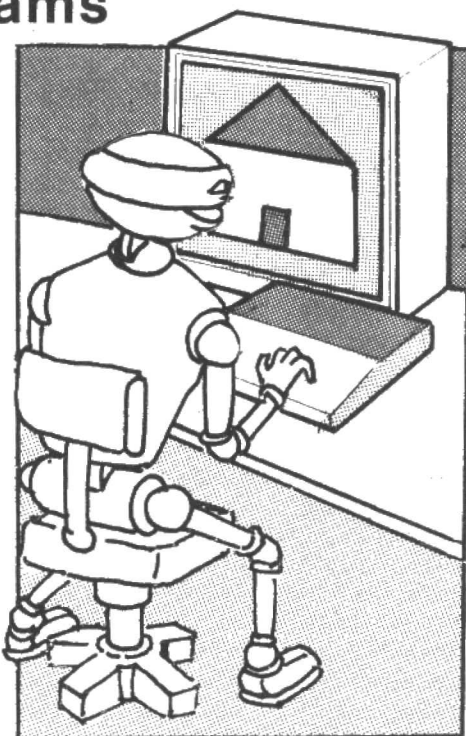
A practical BASIC introduction to intelligent programs

by Mike James

Producing Shapes

The subject of this month's computer brain is one that is not often included in articles or books on artificial intelligence because it seems to have less to do with the way that humans think than other topics. This is a pity because the study of shape and how humans recognise shapes is not only interesting and useful it can also be great fun!

A problem shared by both large and small computers is the production of complicated graphics displays. It's certainly not a problem caused by inadequate hardware. Even some quite small computers can manage eight colours and around 200 by 200 plotting points. You can even find some very powerful software commands in such computers making the drawing of lines, circles, triangles etc. very easy indeed. If you want to see where the problem lies all you have to do is try to use such a system to draw a simple display, say a house with windows and doors. The trouble is that even simple visual scenes are difficult to describe. One way to try to solve the problem of describing shapes is to investigate the way that

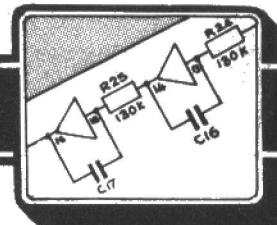


humans and machines tell the difference between different shapes. The reason why the answer to describing shapes is likely to lie in the methods used to tell them apart is that whatever features make a shape different and recognisable are certain to be important in a description of the shape. In the rest of this article the main emphasis will be placed

Fig. 1

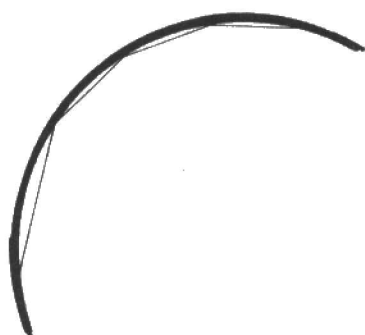


Approximating a curve by straight lines



1). You can improve the accuracy to which a curve is approximated by increasing the number of lines used or by using sections of a more complicated curve. (See Fig. 2). This method is often used on personal computers to build up shapes using a combination of straight lines and circles.

Fig. 2



Improving the approximation by using more lines.

There is an alternative, and more natural way of describing the perimeter of a shape – chain coding. Chain coding is far from new but it has received two recent revivals – in the computer language LOGO and in the graphics commands of machines such as the Dragon. The fundamental idea of chain coding is to specify a starting position, using the usual X, Y co-ordinate system and then describe the perimeter in terms of movements from the current position. For example, a square could be described as –

Start at X,Y

then move 10 units up

then, from your new position, move 10 units to the left

then 10 units down and 10 units right

This recipe for drawing a square is better than an equation because it tells you exactly how to go about drawing a square – it is a list of activities that produce a square. You can work out a chain code for any shape without too much trouble and once you have done it, the description itself forms a program that produces the shape! In addition all this is possible without reference to a co-ordinate system, apart from fixing the starting position of the outline. For all

these reasons, chain coding, an idea that was once the preserve of purely academic artificial intelligence research, has found its way into some popular computer systems. In addition the idea of giving a list of movement instructions to a 'drawing pen', it forms the central idea of the increasingly popular computer language LOGO. (In this case the drawing pen is referred to as a 'turtle' and the list of movement instructions are in terms of turning through a given angle and then moving a given distance).

A chain code program

Users of the Dragon or the Tandy Color computer will already be familiar with the idea of using a chain code to produce outlines. However it is not difficult to write a short program that will add chain coded shapes to any micro with graphics and BASIC! Program 1 is just such a shape drawing program written in Spectrum BASIC. It not only provides a basis for a more complete chain code program, it also serves as an example of how any computer language can be implemented. In other words, this program is in fact an interpreter for an admittedly very simple, graphics language. The program recognises four commands u, d, l and r each of which can be followed by a number. The meaning of the commands is obvious u means up, d means down, l means left and r means right. You can type collections of commands into the program and each will be obeyed in turn. For example, u10r10d10l10 will draw a small square. The way the program works is fairly straightforward and can easily be extended to include other commands. The first part of the program uses subroutine 1000 to initialise any two arrays – a\$(i) holds the letter signifying command i and a(i) holds the line number of the subroutine that has been written to carry out the command. To save space the line numbers are in fact all divided by 1000 so, for example, the subroutine that handles u begins at 3000 not 3. The second part of the program takes the form of a loop that reads in a value for a\$ and splits it down into each instruction using subroutine 2000. The first thing that subroutine 2000 does is to check for a\$ being the null string. Each time that a command is obeyed it is removed from the string and control passes back to the

beginning of subroutine 2000. Thus when all the commands have been obeyed b\$ is empty and control passes back to the main program for another string of commands. Lines 2010 to 2050 try to match the single letter commands against the list of commands in the array a\$. If a match is found then control passes to line 2500. Here the command is removed from the string – line 2500 – and any digits which follow the letter are converted to a number stored in v – line 2600. The digits are also removed from the string so that the next command can be processed in exactly the same way by going back to subroutine 2000. Finally the command is carried out by the correct subroutine being called by line 2520, the value of i indicates which command was found by subroutine 2000 and so a(i)*1000 is the line number of the routine that handles the command.

Program One – Chain Code

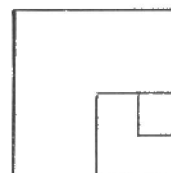
```

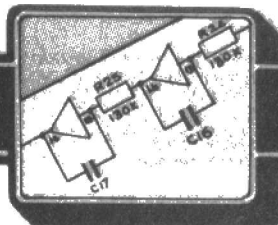
1000 CLS
1010 GO SUB 1000
1020 INPUT b$
1030 GO SUB 2000
1040 GO TO 30
1050 REM command table
1060 DATA "u","d","l","r","s","f","t","b","c","e","a","n","o","p","q","r","s","t","u","v","w","x","y","z","0","1","2","3","4","5","6","7","8","9","."
1070 DIM a$(4)
1080 DIM a(4)
1090 FOR i=1 TO 4
1100 READ a$(i)
1110 READ a(i)
1120 NEXT i
1130 RETURN
1140 IF b$="" THEN RETURN
1150 LET i=1
1160 IF b$(1)=a$(i) THEN GO TO 2
1170 LET i=i+1
1180 IF i>4 THEN RETURN
1190 GO TO 1140
1200 LET b$=b$(2 TO )
1210 LET j=1
1220 IF b$(j)="" OR b$(j)>"9" THEN
1230 GO TO 2500
1240 LET j=j+1
1250 IF j>LEN b$ THEN GO TO 2500
1260 GO TO 1220
1270 LET v=VAL(b$(1 TO j-1))
1280 LET b$=b$(j TO )
1290 GO TO a(i)*1000
1300 DRAW 0,v
1310 GO TO 2000
1320 DRAW 0,-v
1330 GO TO 2000
1340 DRAW -v,0
1350 DRAW v,0
1360 GO TO 2000

```

Sample Output of Program One

u150r100d100l50v50r50d25l25u25





If you look at any of these routines you will see that they are very simple and involve only one BASIC instruction. This is because the Spectrums DRAW command is ideal for carrying out instructions like down 10 in other versions of BASIC you might have to write some quite long subroutines. If you want to expand the program to include commands such as 'mx,y' for move position to x,y, 'rt' for rotate through t degrees etc. then you will find that the subroutines for these are a little longer but the overall method is still the same – identify the command, extract any values and then obey the command.

There are many advantages to using chain codes to describe the outlines of shapes. For example, the same chain code can be used to produce the same shape at a range of sizes simply by multiplying the 'step' following each command by a scale factor. It is left as a practical problem for the reader to work out how a chain code can be used to produce the same shape at a range of different rotations. Another interesting problem is finding out the chain code for a circle – the best way to do this is by experimentation with the chain code program given earlier.

The inner core

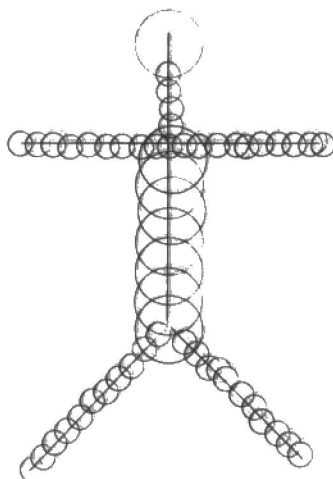
The idea of using the outline or perimeter to describe a shape is so obvious it hardly needs justification. However if you ask someone to sketch a shape like a man the chances are that some of the time you will get a 'stick' man. The use of stick figures can be extended to other shapes without too much trouble. The only real problem is that, whereas the definition of a shape using perimeters was obvious – the shape is the area inside the perimeter – the question "how does a stick figure define a shape?" is less easy to answer.

If you were given the stick figure of a man and asked to fill in the shape what you would do is to thicken up the lines of the figure until things looked approximately right. In other words you would 'flesh out' the stick figure. This is exactly what you have to do to convert a general stick figure into a shape. The only extra information that you need is how much to flesh out the stick figure at each point. In the case of the man figure you use the fact that you already know what a man looks like to supply this information but

in the general case it would have to be supplied with the stick figure. The essence of the idea being explained here is that as well as being able to define a shape by giving details of its boundary you can define a shape by giving details of its 'inner core'. This inner core is sometimes called a skeleton because it defines the shape in much the same way that a human skeleton defines the shape of a human i.e. from the inside!

To actually convert an inner core to a shape we have to be a little more precise about how to flesh it out. One way of doing this is to surround each point of the inner core with a solid disk of a particular radius. The degree of fleshing out would obviously be specified by the radius of the disk. For example, in expanding the stick man to its full shape the disks along the arms would have a smaller radius than the disks around the spine. (See Fig. 3). In general, an inner core is a collection of points each one associated with a radius. To produce the shape defined by an inner core all we have to do is to surround each point by a disk of the specified radius; the shape is just the union of all the disks. Now it might seem to you that it is difficult to define most shapes in this way – what for example, is the inner core of a square or a rectangle? However it can be proved that any shape can be reduced to an inner core and this implies that you can define any shape by using an inner core.

Fig. 3



Expansion of a "man" stick figure.

The details of how to do this are beyond the scope of this article but to give you an idea of how the inner core idea works, program 2 generates random shapes by expanding randomly generated inner cores. Admittedly this program isn't as directly useful as the chain code drawing one but it could easily be modified to allow the user to specify inner cores. The first part of the program calls subroutine 1000 which generates an inner core at random. This is done by storing three random values in the array s(n,3). The variable n determines the number of points in the core and hence the size of the array. For each point there are three entries in the array: s(i,1) is the radius of the disk that should surround the point, s(i,2) is its x co-ordinate and s(i,3) is its y co-ordinate. The second part of the program expands the inner core by calling subroutine 2000. This draws circles of increasing radius around each point of the inner core. This builds up disks around each point until the point's specified radius is reached when it is left out of the circle drawing loop altogether. When the radius of the largest disk is reached the program stops with its final shape.

Program Two – Random Inner Cores

```

10 GO SUB 1000
20 LET r=0
30 GO SUB 2000
40 STOP
1000 LET n=INT
  (RND*10)+5
1010 DIM s(n,3)
1020 FOR i=1 TO n
1030 LET s(i,1)=INT
  (RND*50)+1
1040 LET s(i,2)=INT
  (RND*50)+50
1050 LET s(i,3)=INT
  (RND*50)+50
1060 NEXT i
1070 RETURN
2000 LET r=r+.5
2010 LET f=0
2020 FOR i=1 TO n
2030 IF s(i,1)=0
  THEN GO TO 2070
2040 CIRCLE s(i,2)
  ,s(i,3),r
2050 LET s(i,1)=s
  (i,1)-1
2060 LET f=1
2070 NEXT i
2080 IF f=1 THEN GO
  TO 2000
    
```

Sample Output
from Program
Two



You can have many happy hours watching the randomly generated shapes being built up from the inner cores. However there are a few serious points to be learned from the program. Firstly some of the points of the randomly generated inner core don't affect the shape. The reason for this is that the radius of the disk that surrounds them is smaller than a nearby disk which swallows them up. Such points can be dropped from the inner core without altering the shape of the final object. An inner core that has no redundant points in it is clearly the most efficient for reconstructing an image. You might like to consider how you could remove redundant points from a random inner core before you started to expand it. The second point is that the fine detail of shapes comes from points far away from the centre of the object and with small disks surrounding them. You can go on making observations like this for quite some time. For although we are very used to seeing how a shape is defined by a perimeter we are very inexperienced at seeing how the inner core effects the shape of the object that it defines.

For further work and programs you might like to change the random inner

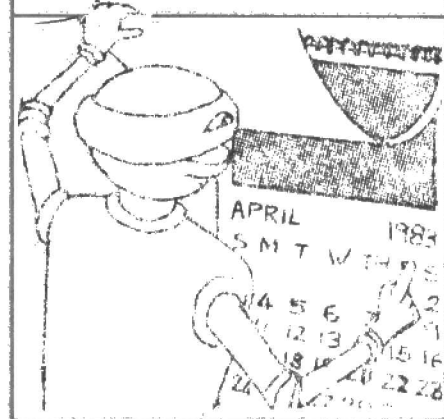
core program to allow you to input the points of an inner core from the keyboard or via a joystick. Then you could try to solve the following two problems. First, what is the inner core of a circle. Secondly, what is the inner core of a square? As a much more difficult problem you might like to write a program that will 'strip down' a shape to its inner core. If you have found out how to remove the redundant points by modifying the random inner core program you could also produce the inner core with the smallest number of total points.

Conclusion



This month's computer brain has been something of an excursion into the realms of geometry rather than A.I. However it shouldn't be forgotten that most of the ideas concerning chain codes and inner cores come directly from A.I. research. Chain codes have been used in computer recognition of shapes for some years. The inner core is not only useful in machine shape recognition there is a suggestion that it might correspond to the method that humans use to recognise shape!

Next month Computers that think?



DESTROYERS

(16 & 48K ZX Spectrum) New and original arcade game in real time. Written in high speed machine code with hi-res graphics, full colour and sound effects. Test your skill against the awesome and varying firepower of the different waves of Destroyers. High score saved **£6.50**

ORB

(48K — Spectrum; 16K — Vic 20; Dragon 32) Make your way through the underground labyrinth in your search for the dreaded Orb, which you must destroy. Encounter many Monsters, discover Treasure and try to remember your route so that you can get out again. Full sound effects and save game facility **Only £5.00**

STAR FIGHTER

(16K — Spectrum) All action, full-colour, graphic machine-code. Space-battle with devastating explosions. On screen scoring and high score kept. The longer you survive the more difficult it becomes. **Only £5.00**

GAMES PACK (Unexpanded Vic 20) Alien, Road Race, The Island, Pontoon. **Only £5.00**

ALL ORDERS DESPATCHED BY RETURN

I enclose a Cheque/PO for £

Name _____

Address _____

Post Code _____

All prices include P&P and VAT

ECM05

IMPACT SOFTWARE
70 Redford Avenue
EDINBURGH EH13 0BW
TEL 031-441-4257

Please Supply:

Destroyers (£6.50) ☐ Star Trek (£5.00) ☐

ZX Trek (£6.50) ☐ 3-D Maze (£5.00) ☐

The Quest (£5.00) ☐ Starfighter (£5.00) ☐

Orb (£5.00) ☐ Games Pack (£5.00) ☐

Please state machine type _____

ZX TREK

(48K ZX Spectrum) First quality star trek game in real time with hi-res graphics plus constant on screen display of galaxy map, long range scan, and status report. Over twenty commands with full colour and sound effects. This game provides a real challenge for the ZX Spectrum game player **£6.50**

STAR TREK

(16K — Spectrum; 8K — Vic 20; Dragon 32) Save the Galaxy from the Klingons using your rapid-fire phasers and photon torpedoes. Automatic short-range scan, Galaxy map and Star-bases. Full sound effects and 10 levels of difficulty! **Only £5.00**

NEW! 3-D MAZE

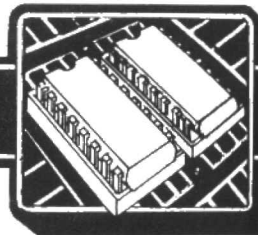
(48K — Spectrum; Dragon 32) Exciting 3-Dimensional Maze Game! Search for the 3 fabulous treasures, then make your way back to the exit. Time yourself with the On Screen Clock — But Beware — the treasures are not always in the same locations **Only £5.00**

Dealers — Attractive Discounts
Spectrum & VIC 20 programmes
wanted — 25% Royalties Paid.

THE QUEST

(48K — Spectrum; Dragon 32) One of the most exciting adventure games currently available. Fight your way into the depths of the complex in your Quest for the Holy Grail. Discover Gold and Precious stones, buy weapons and Magic wares from a trader. Battle with one of the many Monsters. Up to 1500 locations may be searched in the course of a game. Full sound effects and save game facility **Only £5.00**





JUPITER ACE ELECTRONIC SCRAMBLER

by *Ralph Hilton*

This article details the listing and theory behind a program that enables one to scramble screens of data in such a fashion that they can only be decoded by use of a specific codeword. This is not a simple substitution code that can be broken by studying the frequencies of symbols but one that uses a bit by bit coding available with the use of the exclusive or function of the Ace.



First of all I shall go into the logic methods of the coding technique then, as FORTH is a new language for a lot of readers, I shall explain the listing step by step.

The basic XOR function in the Ace compares 2 bytes of memory bit by bit returning a 1 whenever the 2 bits compared are different. For example (in base 2) 11110000 10101010 xor gives 01011010. It is an interesting and useful feature of the xor function that if you perform the operation twice with the same number then you get back what you started with so that x y xor y xor gives y. In our earlier example 01011010 10101010 xor gives us back the original 11110000.

As each character in the memory of the computer is held as an 8 bit binary number it is fairly simple to hold a code

word in memory and xor a message to be coded with it. This gives us a scrambled message that is totally lacking any pattern and will not yield to any attempts at frequency analysis. To crack this form of coding would, I imagine, require a considerable time from a large computer which would have to analyze the billions of possible combinations in a long and laborious fashion. While it is not totally undecodable it is probably safe from anything but a large mainframe with some pretty intricate software.

The program itself performs all its coding and decoding operations with the display file so that one puts the message to be coded or decoded onto the screen and the result appears there. This makes it very simple to store as the saving facilities of the Ace allow one to save a series of bytes. With the program loaded the coding operations can be performed in a matter of seconds. 224 bytes of data can be coded at one time and with each screen being saved as it is coded there is no real limit to the capacity.

I shall give each word of the listing and explain what it does as I go along. Comments in brackets should not be typed in.

: DISP (Puts your message into the display file area)

CLS." Type in your message of up to 7 lines ending with !"

QUERY ASCII !WORD DUP 1 + SWAP C@ INVIS CLS TYPE ;

The word WORD is explained on page 97 of the Ace manual and should make clear how the above works.

: Z INVIS 21 0 AT ." Enter number of complete lines of message"

21 SPACES QUERY NUMBER DROP 32 *

21 0 AT ." Enter number of characters in any partial line or 0 if none"
QUERY NUMBER DROP + DUP
224 > IF DROP 224 THEN

21 0 AT ." Enter code word up to 63 characters with no spaces"

16 SPACES QUERY 32 WORD DROP

9216 + 9216 ;

First of all this word is used by the code and decode functions so it written separately and called by each to save memory.

It asks for the exact length of your message. This is done as we do not want to code a lot of spaces at the end because this would give away the code word.

It then asks for the code word and puts it into a section of memory called the PAD. This is essentially a workspace used for character and string handling by the Ace.

The 2 numbers left on the stack are the parameters for the code and decode functions telling them which section of the screen to code/decode.

: CODE Z 7 0 AT 32 SPACES

DO

IC@ I 9216 - PAD C@ MOD

PAD «+ + C@

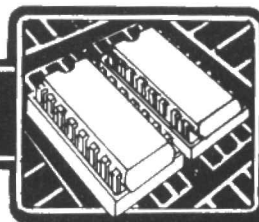
I 3 MOD 128 * +

XOR 91 +

I 256 + C!

LOOP ;

This takes a character from the top section of the screen and XORs it with a character from the code word then prints it 8 lines further down. The sections of coding I 3 MOD 128 * + and 91 + are just to add complexity to the procedure and avoid any repetitions which could reveal the code. These become apparent if you change the words to leave them out.



```
: DECODE Z 7 0 AT 32 SPACES
DO
  I C@ 91 — I 9216 — PAD C@
  MOD
  PAD 1+ + C@
  I 3 MOD 128 * +
  XOR
  I 256 + C!
  LOOP ;
```

This is the reversal procedure similar to the code routine except for the positioning of the 91.

It takes a message from the top of the screen and decodes it into the section 8 lines down.

```
: TRANS
  0 0 AT 8448 224 TYPE ;
```

This just transfers the lower section of the screen to the top part used mainly for testing your program.

```
: CLRC
  0 0 AT 224 SPACES ;
```

```
: CLRD
  0 0 AT 224 SPACES ;
```

These 2 words clear the upper and lower sections of the screen as needed. Next, to simplify the save and load procedures, the next words allow saving from the coded section of the screen and loading into the upper part of the screen. They are used just like LOAD and SAVE.

```
: CSAVE 8448 224 BSAVE ;
```

```
: CLOAD 8192 224 BLOAD ;
```

That completes the listing. You can now use the program by entering DISP then entering your message. Entering

CODE and then following the instructions on the screen enable you to get the coded version of the text. If you want to decode it again then move it to the top of the screen with TRANS then enter DECODE and follow the instructions on the screen.

A coded message is saved by entering CSAVE messagename. Reloading with CLOAD will put it into the top section of the screen ready for decoding.

To ensure that one's coded message is secure you should be careful with the exact length of the message being put in correctly. The degree of security provided rises exponentially as the length of the code word. If you use a single letter code then it would only take a few hours to crack whereas a code word as long as the message itself would make it totally impossible.

COMPUTER TRAINING

FULL TIME COLLEGE COURSE

SUITABLE FOR APPLICANTS WHO WISH TO ENTER COMPUTER SERVICE OR RELATED INDUSTRIES — HIGH PERCENTAGE OF PRACTICAL COURSE WORK

15 MONTHS

TEC Certificate in Computing Technology

6 MONTHS

TEC Higher Certificate in Computing Technology

Subjects: Foundation Electronics, Digital Techniques, Microelectronics, Microprocessors, Microcomputer Based Systems, Industrial Robotics, Machine Code & High Level Programming.

Shortened courses can be arranged for applicants with previous knowledge.

Courses commence April, September & January.

Prospectus from:

ECM05

LONDON ELECTRONICS COLLEGE
20 Penywern Road,
Earls Court, London SW5 9SU.
Tel: 01-373 8721

NEEDS NO EXAMINATION

You don't need to look too closely at the GSC Proto-Clip range of IC test clips to see that they provide instant connection to dual-inline packaged components. Use them on ICs, networks or relays to provide a high-integrity interface for 'hands-off' testing with oscilloscopes, signal sources, logic analysers and other instruments. Contact spacings designed to suit all standard IC packages, GSC Proto-Clips feature a moulded webhinge construction, non-corroding nickel-silver contacts, and clip notches to prevent slippage during testing. Make contact with GSC Proto-Clips right away by filling in the coupon.

GSC (UK) Ltd. Unit 1, Shire Hill Industrial Estate, Saffron Walden, Essex CB11 3AQ Telephone: (0799) 21682 Telex: 817477

For larger quantities, direct price offers and other enquiries, call 0799 21682.

14 Pin PC14 £3.25 Retail £14.80	16 Pin PC16 £3.45 Retail £4.83	24 Pin PC24 £6.00 Retail £8.06	40 Pin PC40 £19.35 Retail £11.90
------------------------------------	-----------------------------------	-----------------------------------	-------------------------------------

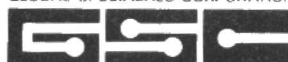
Bold prices include P & P and 15% VAT

I enclose cheque/PO for £
or debit my Barclaycard, Access, American Express card
No Exp. date
or Tel: (0799) 21682 with your card number and your order
will be in the post immediately.

NAME

ADDRESS

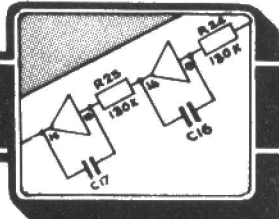
GLOBAL SPECIALTIES CORPORATION



FREE catalogue tick box []

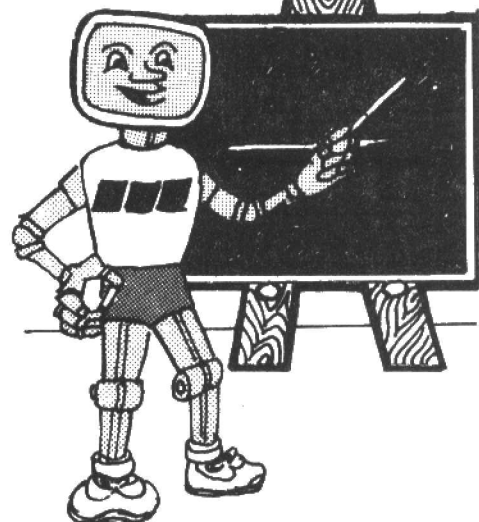
Global Specialties Corporation (UK) Limited, Dept. 340Q
Unit 1, Shire Hill Industrial Estate, Saffron Walden, Essex CB11 3AQ.

TOMORROW'S TOOLS TODAY



COMPUTING EXPLAINED

Article 3 by B. Boyde-Shaw



Graphics

Most home computer manufacturers these days advertise the fact that their machines have a graphics facility plus full colour and sound. In this article we will look at colour and graphics and how the computer carries out the commands concerned.

Graphics characters are in fact no different from text characters, except that the character has been coded differently.

The method for coding the capital letter T, for example, is exactly the same as for coding a graphics character.

Using the matrix we had in a previous article we must first plan what our graphics character is to look like. For example fig. 1, our little dog Fred has somehow to be put into a program so that we can do things with him on the screen.

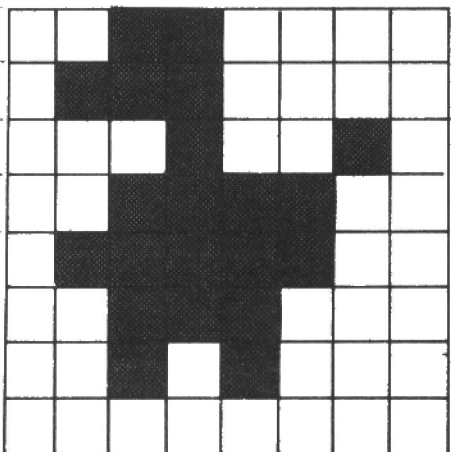


Fig. 1

matrix

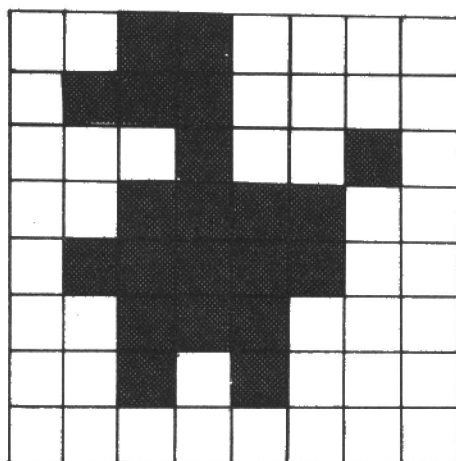


Fig. 2

Many computers have some means of re-defining new characters or filling empty ones. Normally in BASIC there are 256 characters available, more often than not all definable by some digitised code, using either a binary or hexadecimal system, this is, using either a 1 or a 0 (binary), or 0-9 plus A-F (hexadecimal) to count with. For example, in binary, 00001111 means 15 and so does F in hexadecimal, or 00001000 in binary means 8 in decimal and 8 in hexadecimal, understand? No? The table in fig. 3 will help. So to put our 8 x 8 matrix dog into the computer we must first code our character, for example, one method could be as in fig. 2.

Now we must put this information into our program, and to do this we tell the computer to define the character of our choice to represent Fred.

binary

decimal

= 00110000 =	48
= 01110000 =	112
= 00010010 =	18
= 00111100 =	60
= 01111100 =	124
= 00111000 =	56
= 01010000 =	80
= 00000000 =	0

There are many ways of doing this depending on the computer, for example BBC Basic uses "VDU 23", Texas uses "CALL CHAR", Sinclair uses a "BIN" command. So let's build up a program for a fictitious computer to give you an idea of how it's done.

5 CLS (means clear the screen memory)
10 (Define Fred command) CHR\$(240),48,160,18,60,124,56,80,0

What we have done is to tell the computer to make character no. 240 using the decimal total for each line of the Fred matrix in order, from top to bottom. As you can see from Fig. 3, the matrix you can use starts at the right hand side with a 1 and doubles for each square to the left up to 128.

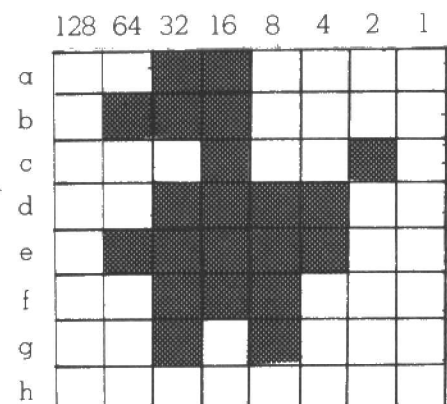
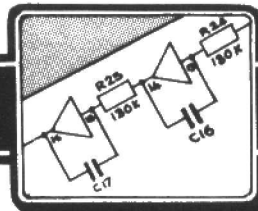


Fig. 3

Then each line is added up using those squares which are filled in.

$$\text{row a } 32 + 16 = 48$$

$$\text{row b } 64 + 32 + 16 = 112$$

$$\text{row c } 16 + 2 = 18$$

etc.

Then each total is placed in CHR\$(character string) in turn from a to h. If you look at the binary column in Fig. 2, you will see that each filled square corresponds to a 1, and each empty one to 0, (this is called binary arithmetic).

If you put these 2 lines into our computer now, you wouldn't see anything happen when you ran the program. That's right you guessed it, we haven't used a PRINT statement yet.

Now to position our dog where we want it on the screen we must use a special PRINT statement, namely PRINT TAB or PRINT @ or similar. This locates the character in a given position depending on the row and column layout on your TV screen.

For example, if your screen can show 24 rows by 32 columns, to place the dog in the middle of the screen we would type in PRINT TAB (16,12);CHR\$(240) which translated into English means put our dog 16 columns across the 12 rows down on the screen. So put this in our program.

35 PRINT TAB (16,12);CHR\$(240)
But of course Fred will be hanging in mid air so let's give him a pavement to walk on, so add a few lines:

20 FOR P = 1 TO 32

25 PRINT TAB (P,13);CHR\$(241)

30 NEXT P

The FOR-TO-NEXT loop here repeats CHR\$(241), 32 times across the screen. What is CHR\$(241) - maybe nothing yet, so we must define CHR\$(241) to be

a full block by (see Fig. 4)

15 (Define Block command)
CHR\$(241),255,255,255,255,255,
255,255,255

It is possible you would already have in your BASIC a full block character, so if you had you would use that CHR\$ instead of defining one.

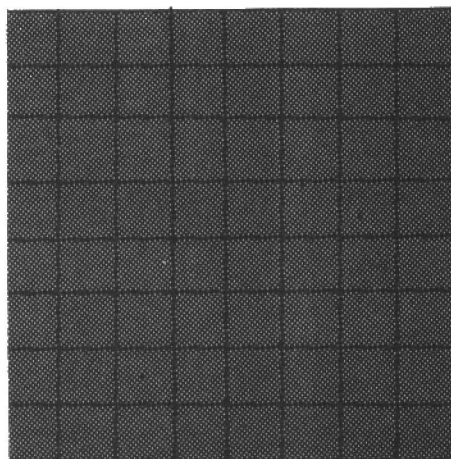


Fig. 4

Fred is now standing on the pavement but we want him to move, so if we remove him from position 16,12 and then replace him at position 15,12, he would seem to jump to the left one character space.

45 PRINT TAB (15,12);CHR\$(240)

50 PRINT TAB (16,12);CHR\$(32)

CHR\$(32) is a space, i.e. an empty matrix, could you define it?

and again

60 PRINT TAB (14,12);CHR\$(240)

65 PRINT TAB (15,12); CHR\$(32)

and so on till he disappears off the edge of the screen. How about making him jump up and down, or re-defining him and making him lie down.

Of course this all takes place quite quickly, depending on the speed of operation of your computer, so we must delay the position changing routine by putting a delay sequence in as follows:

1000 FOR D = 1 TO 3000

1005 NEXT D

1010 RETURN

This is the same as the previous FOR-TO-NEXT statement but with nothing happening, i.e. just a delay. Note the TO number must be a lot bigger.

And then using this as a sub routine, that is, going to this part of the program only when we need to. So let's list the whole program and see what we have got.

5 CLS

10 Define FRED as CHR\$(240)

15 Define pavement block as
CHR\$(241)

20 FOR P = 1 TO 32

25 PRINT TAB (P,13);
CHR\$(241)

30 NEXT P

35 PRINT TAB (16,12);
CHR\$(240)

40 GOSUB 1000

45 PRINT TAB (15,12);
CHR\$(240)

50 PRINT TAB (16,12);
CHR\$(32)

55 GOSUB 1000

60 PRINT TAB (14,12);
CHR\$(240)

65 PRINT TAB (15,12);
CHR\$(32)

70 GOSUB 1000

1000 FOR D = 1 TO 3000

1005 NEXT D

1010 RETURN



Fred now moves along the pavement a few steps.

And next we can add some text into our picture, so we could type in

75 PRINT "BACK, FRED"

80 GOSUB 1000

Then define an extra character for FRED going the other way as in Fig. 5, and make him go back along the pavement.

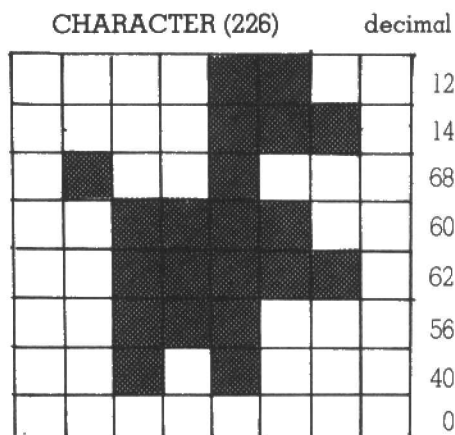
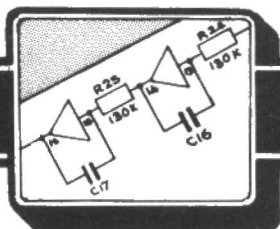


Fig. 5



```
85 (Define 2nd Fred command)
  CHR$(242),12,14,68,60,62,56,
  40,0
90 PRINT TAB (14,12);
  CHR$(242)
95 PRINT TAB (13,12);CHR$(32)
100 PRINT TAB (15,12);
  CHR$(240)
105 PRINT TAB(14,12);CHR$(32)
```

Now see if you can move him across to the right hand side of the screen and then make him disappear, using the correct commands for your own computer. If you haven't got a computer then use my fictitious one.

The PRINT TAB statement is a useful device but there are many other statements used in graphics programs, for example:

POKE, PLOT, MOVE, DRAW, FIL, CIRCLE, ETC.

which can all be a whole lot quicker too. Of course we have only used character graphics in our program and not high resolution graphics, where any one of those 64 squares, called pixels, in an 8 x 8 matrix, can be used and positioned individually. High resolution graphics is an exciting and absorbing hobby in its own right and deserves a series of articles on its own. (How about it Ed?).

Now to colour. In our program Fred would no doubt have been white, on a black screen, or whatever default combination your computer used; default means the unprogrammed condition that your computer operates in.

To give Fred a colour requires your characters to be colour coded via your program.

Some computers, for example, the Texas TI99/4A, allows you to colour a given set of ASCII characters in one colour, and not each individual character on its own. For example set 5 is capital letters A to G and the @ sign. This is a bit restrictive, in that you must be careful which character you choose to re-define as the dog and the pavement, otherwise they would always both be the same colour.

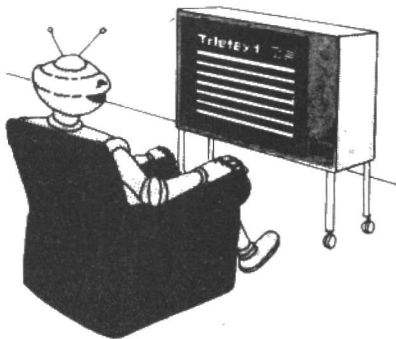
Other computers only allow you to use a given number of colours at any one time.

The command used is usually of the type that indicates what it is doing, for example "INK", or "COLOUR", which colours what you are writing with, i.e. the character. (PAPER is often used to colour the background too, or

COLOUR plus a large number for the background).

So we must now colour the pavement by adding the correct command to our program at the correct time and then colour Fred in with another command. For example in BBC BASIC you would use the correct colour command just prior to printing Fred, and colour and print the pavement at the beginning of the program.

Any program however small is enhanced by adding some colour. Just turn down the colour control on your television set, and notice the difference in quality of the programme; be it a written screen full such as a Teletext broadcast or a colourful 'London Night Out' programme.



The extra effort involved in including not only colour but different colours in a program is well worth while. It's an interesting study on its own to experiment with different coloured dogs and pavements and see which colours go better together and which do not.

Even in a black and white program, for example a textual quiz, colour can be used, one for the question, one for the answer. Other important parts of a program can be coloured to make them stand out on the screen. One step on from this would be to also change the colour of the background on which the words are written. In our small program we wrote on the screen BACK FRED, we could change this using a colour for the text with a different coloured background, with the INK and PAPER, or COLOUR commands.

Excellent you say, but I've only got a Black and White television. Never mind, with a minimum of 8 colours on your computer you can usually get a good range of grey scales but you would have to get the brightness and contrast controls set up to suit your program. Experiment with the colour commands

and make a note of those that give the biggest contrast between each other, starting at black through to white. One way is to make up a grey scale program for your computer. For example:

```
5 CLS
```

```
10 Define a block character (or use
  one you have in your ASCII codes)
20 FOR I = 1 to 8 (if you have 8
  colours available)
```

```
25 COLOUR I
```

```
30 PRINT CHR$(241)
```

```
35 NEXT I
```

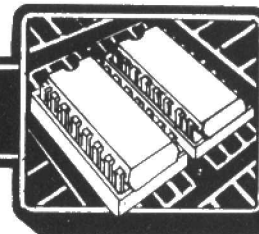
Now change your background with a PAPER or other colour command. Go on, experiment with the commands we have talked about in this article, and if you have a problem you can't solve contact me through the magazine or at 7 Riverway, Nailsea, Avon.

The world of computing is infinitely large and has many strings to its bow, but all lead eventually to a program run of some sort. Only attention to detail and good practice in writing will lead to a good program, which, after all, is what these articles are all about.

Next time we will deal with Speech and Music and perhaps get Fred to bark and scratch his claws on the pavement.



**REMEMBER TO
ASK YOUR
NEWSAGENT
FOR
ELECTRONICS
& COMPUTING
MONTHLY**



INTERFACING THE DRAGON 32 MICROCOMPUTER

by A. G. Nanson

INTRODUCTION

Memory Map and Expansion

On looking at the Dragon 32 memory map (see the Additional Information booklet supplied with the machine), it can be seen that the input/output area extends from address HFF00 to HFF5F. Some of this area is occupied by the on board MC6821 PIA's which are responsible, amongst other things, for interfacing the keyboard, the video display generator, the cassette and printer, etc. However, addresses HFF40 to HFF5F are available for external use.

The Dragon 32 expansion port consists of an unpolarised 20 x 20 way 0.1" pitch edge connector situated in the recess on the right hand side of the machine, i.e. the ROM cartridge port. The pin numbers of this edge connector are shown in Fig. 1 and the Pin Out in Table 1.

Table 1
Pin out of Edge Connector

Pin No.	Function	Pin No.	Function
1	+12V	2	+12V
3	HALT	4	NMI
5	RESET	6	EIN
7	QIN	8	CART
9	+5V	10	D0
11	D1	12	D2
13	D3	14	D4
15	D5	16	D6
17	D7	18	R/W
19	A0	20	A1
21	A2	22	A3
23	A4	24	A5
25	A6	26	A7
27	A8	28	A9
29	A10	30	A11
31	A12	32	R2
33	GROUND	34	GROUND
35	SND	36	P2
37	A13	38	A14
39	A15	40	EXT. MEM

Choice of Interfacing Chip

The obvious interface chip to use with the Dragon 32 is, of course, the

MC6821 Parallel Interface Adaptor (PIA). However, as a SY6522 Versatile Interface Adaptor (VIA) chip was to hand, it was decided to use this instead. Although the 6522 VIA is a somewhat more expensive device than the 6821 PIA, it does have more functions; a fact which could be of interest to the experimenter.

For example, in addition to the two 8 bit bi-directional input/output ports and their associated control lines, the 6522 VIA also contains two programmable timers, and a shift register. As the 6522 VIA is a more complex device to program—it has 16 addressable registers compared with the MC6821's 6 addressable registers—the user is recommended to obtain a copy of the data sheets before using.

There is, of course, no reason why a MC6821 PIA should not be used. In fact, it may well be thought that to use a 6522 VIA for the simple interfaces described in these notes, is to 'gild the lily'. With this in mind some notes will be included on the use of the MC6821 PIA.

Simple Interfaces

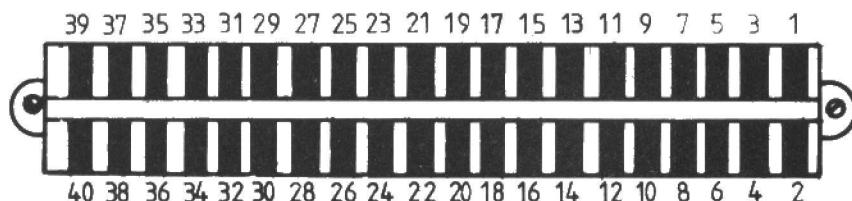
Figure 2 shows a simple parallel interface using a 6522 VIA. Port B of the VIA is used in the input mode in order to read the state of the eight toggle switches SW1 – SW8. Open switches are equivalent to logic 1. Closed switches are equivalent to logic 0. Any number between 00000000 and 11111111 (0-255) can be thus represented.

Port A is configured in the output mode to drive the eight LED's D1-D8 connected in the common anode mode. The 74LS05's acting as current sinks. A logic 1 output on any of the A port lines will be inverted by the 74LS05 causing the corresponding diode to light.

Continued on Page 54

ELECTRONICS & COMPUTING — 51

Fig. 1 DRAGON EDGE CONNECTOR



Continued from Page 51

Fig. 2

A SIMPLE PARALLEL INTERFACE FOR THE DRAGON 32

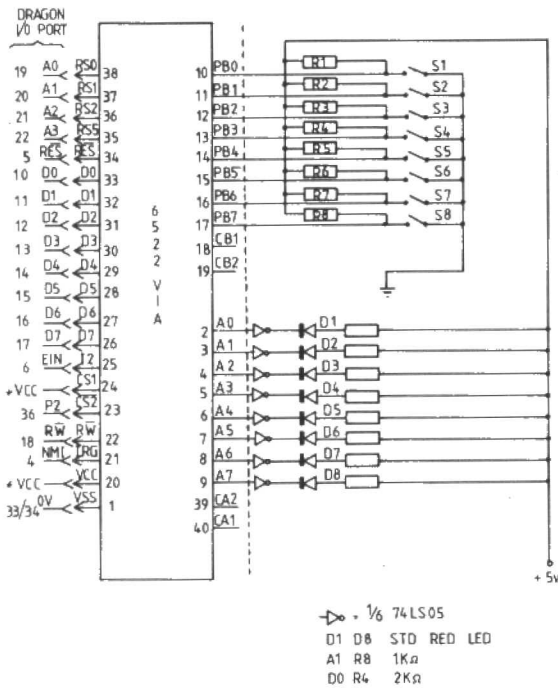


Fig. 3

SIMPLE METHOD OF BUFFERING 6522 VIA

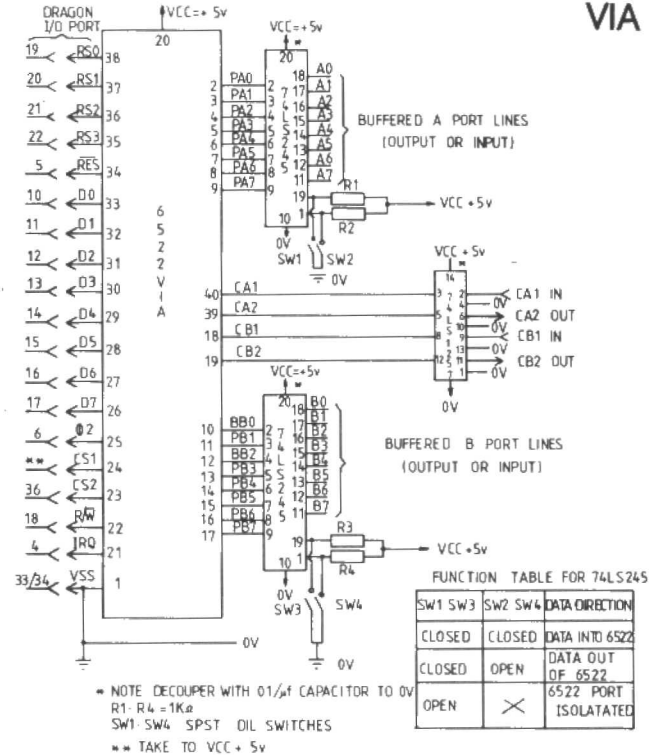


Figure 3 shows a modified version of the interface. The A and B port lines are buffered by 74LS245 octal bus transceivers. For the sake of simplicity the data direction through the 74LS245's is determined by manually setting the switches SW1-SW4, according to the function table.

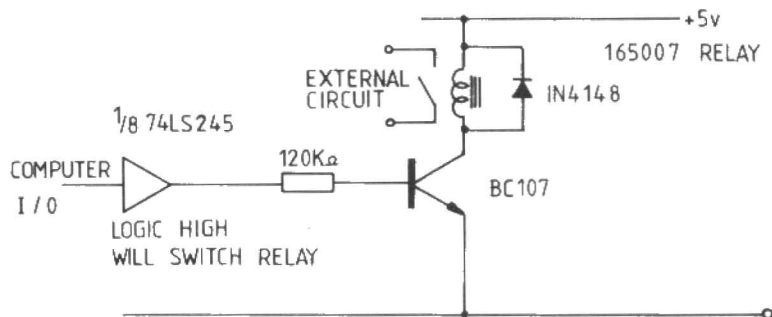
The port control lines CA1, CA2, CB1 and CB2 are protected by a 74LS125 quad buffer. Lines CA1 and CB1 are configured for input only; lines CA2 and CB2 for output only. Naturally, some sacrifice in flexibility has been made by buffering the 6522 VIA in this manner.

Connecting the Interface to External Devices

A simple method of using the computer to control external devices is by having it operate suitable relays. For low voltage/low current applications it is possible to use TTL compatible DIL reed relays or CMOS analogue switches. Both of these can normally be driven directly from the

Fig. 4

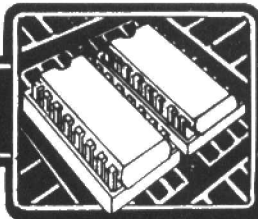
SIMPLE TRANSISTOR SWITCH FOR SMALL RELAY



computer interface. However, if it is required to switch high voltages/currents, it is necessary to use a heavier relay. It is usual to operate these relays by means of a switching transistor which is turned on or off by the computer.

Figures 4 and 5 are examples of such

circuits, the former being useful in relatively light applications, whereas the latter circuit should be employed when it is required to switch high voltages; the computer being electrically isolated from the switched circuit by the opto-isolator.



Constructional Notes

Various methods are available to the constructor for making up the interfaces described. For experimental work the use of prototyping breadboards or plugboards is to be preferred, for although the initial cost of these is quite high, in the long term they are extremely good value, when one considers with what ease complex circuits can be assembled or modified. More permanent circuitry may be assembled on Vero V-Q board (no track cutting needed). However, it is advisable to use a good quality temperature controlled soldering iron and to have a desoldering tool available.

Whatever methods of construction are employed, it is advisable to handle the PIA or VIA chips with some care. When handling these chips it is preferable to work on a grounded metal surface – a stainless steel draining board for example. It is also wise to socket all chips and to insert them only after the circuit has been thoroughly checked and **before** any power is applied.

Power Supplies

Rather than use the computer power supply, it is preferable to have a separate 5 volt regulated supply for the interface. For quite apart from the question of overloading the on board voltage regulator, with consequent over-heating problems, any short circuits should only result in the demise of the interface and not the computer.

If a suitable 5 regulated supply is not already to hand, the components for a simple 5 volt 1 amp regulated PSV, including the PCB and heat sink, can be obtained from Maplin Electronic Supplies.

Connecting the Interface to The Dragon 32

Connecting the interface to the Dragon's I/O port may pose a problem. Some form of 20 x 20 way x 0.1" pitch insert is required. However, the computer's I/O port edge connector socket is somewhat difficult to get at, hiding coyly as it does at the end of a deep and dark recess, and

Fig. 5

SUGGESTED METHOD FOR ISOLATING SWITCHING CIRCUIT FROM COMPUTER

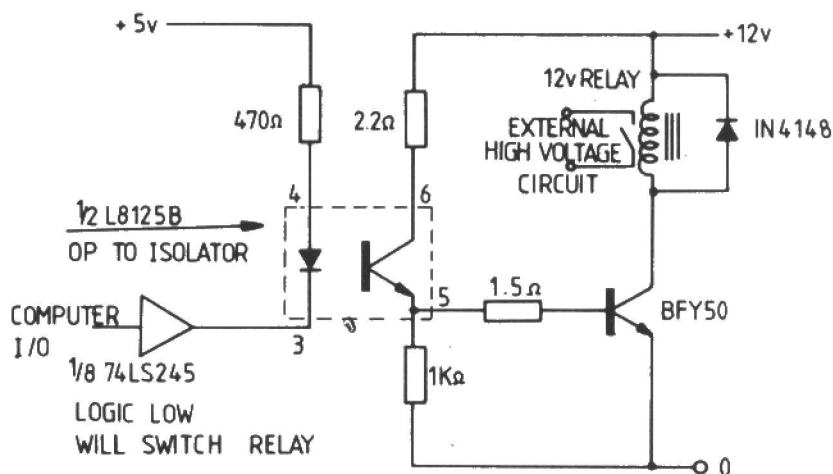
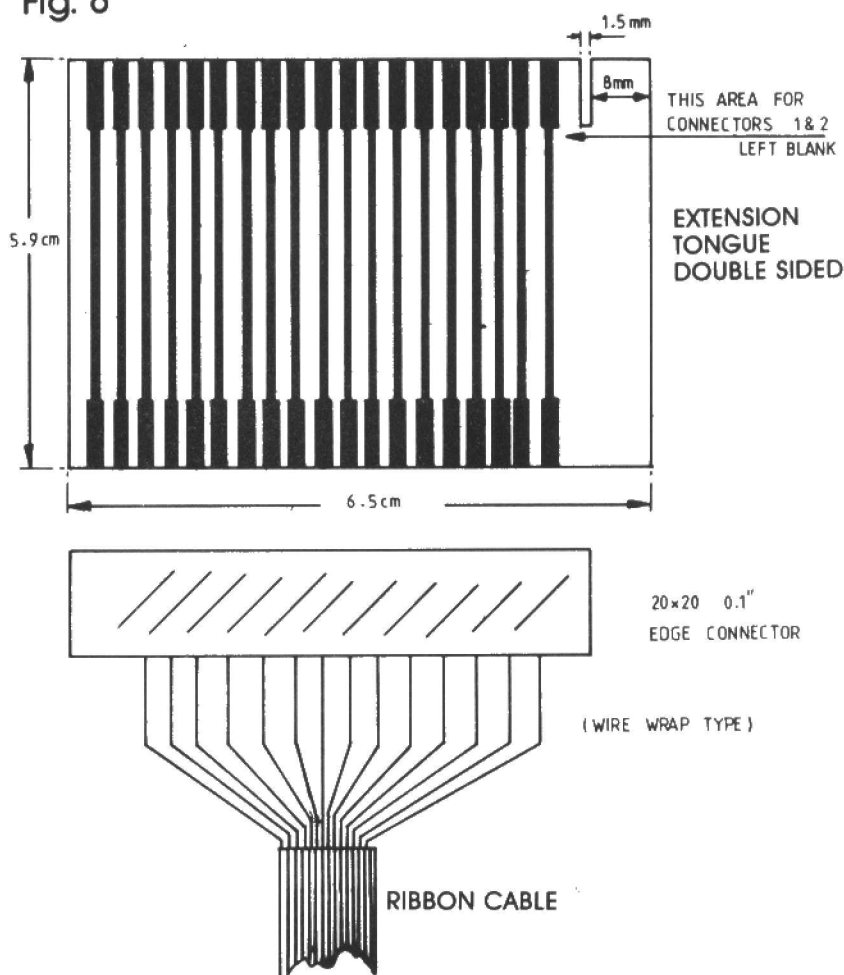
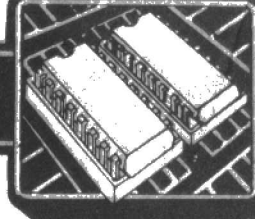


Fig. 6





obviously care must be taken that nothing is damaged or shorted out when the interface is connected.

Figures 6 and 7 illustrate the method used by the author to connect his interface to the Dragon. The extension tongue is made from a piece of double sided PCB, cut to the dimensions specified. The tracks were laid out with the aid of acid resistant transfers (obtainable from Maplin Electronic Supplies).

Great care must be taken to ensure that the tracks on each side are correctly aligned before attempting to etch, for the success of the tongue depends upon the correct alignment and correspondence of the tracks on each side of the PCB.

After the board has been etched the tracks should be cleaned, and after drying, carefully tinned. If possible the whole tongue with the exception of the contact area, should be given a coat of insulation varnish, after completion.

It will be noticed that the 12 volt tracks have been missed out. This is intentional, for a short from one of these to certain of the other tracks could prove to be a source of upset to both the Dragon 32 and its owner.

Programming The Interface

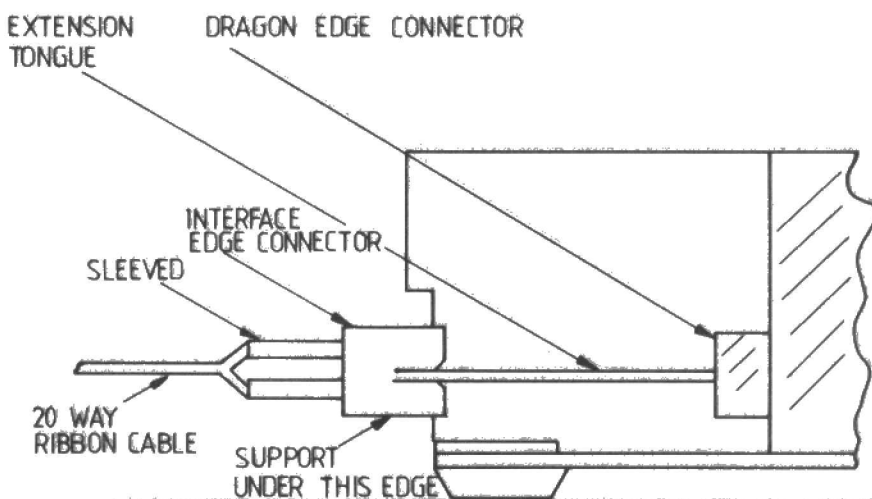
As previously mentioned, the 6522 VIA is a somewhat complicated device, and to make full use of its available functions it is necessary to have the data sheets. However, if these are not available, the following information should be sufficient to enable the builder to operate the simple interfaces dealt with in the article.

With the interface connected as described, we will be concerned with addresses in the area HFF40-HFF4F; and of the 16 VIA registers, we will be interested in the following:-

Register Number	Register Designation	Description	Dragon Interface Address
0	ORB/IRB	Output Register "B" Input Register "B"	H FF40
1	ORA/IRA	Output Register "A" Input Register "A"	H FF41
2	DDRB	Data Direction Register "B"	H FF42
3	DDRA	Data Direction Register "A"	H FF43
12	PCR	Peripheral Control Register	H FF4C
14	IER	Interrupt Enable Register	H FF4E

Fig. 7

CROSS SECTION THROUGH DRAGON CARTRIDGE PORT SHOWING METHOD OF CONNECTING INTERFACE



To operate the interface in the 8 bit parallel input or output mode, adopt the following procedure. (Note: If the ports are buffered as described (Fig. 3) all bit in that particular port must obviously be set in the same mode).

Port A — Input Mode

POKE HFF43,0

(This will set DDRA in the input mode)

PEEK (HFF41)

(To read Data on IRA)

Port A — Output Mode

POKE HFF43,255

(This will set DDRA in the output mode)

POKE HFF41, DATA

(DATA OUT ON ORA)

Port B — Input Mode

POKE HFF42,0

(This sets DDRB in the input mode)

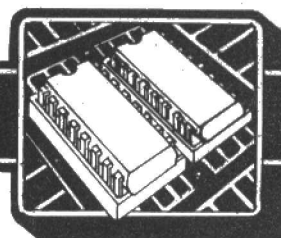
PEEK (HFF40)

(To read Data on IRB)

Port B — Output Mode

POKE HFF42,255

POKE HFF40, DATA



Program I

This is a simple Input/Output driver for the interface. It is basically a polling routine where the value, according to the setting of the switches connected to the B port, is read. This value is then sent to the A port causing the LED's to light, and also printed to the screen.

If the switches are set to a pre-determined value (chosen in line 70), the program jumps to a simple routine which sounds an alarm for a few seconds, after which the program may be returned to the main loop by pressing any key.

Lines 10-110 form the main section of the program, initialising the ports, reading the status of B and out-putting to A and to the screen.

Lines 130-140 clear the screen, once it has filled, but retain the title.

Lines 150-230 comprise the alarm routine, giving a visual and audio indication.

Program II

This program is perhaps a little more interesting, if not in its content, perhaps in its implications. It consists primarily of two machine code routines (a main routine and an interrupt service routine (ISR)), plus a basic loader program.

The 6522 VIA IRQ pin is connected to the Dragon's NMI line (pin 4). In this case a non-maskable interrupt (NMI) is produced when the CA1 line of the 6522 is pulled momentarily low (See Fig. 8).

The effect of the NMI is to cause the 6809 CPU to look immediately at the instruction stored in the NMI Vector (Addresses H109-H10B) and act upon it. In program II the instruction stored is a jump to address H7F8B, the start of a simple ISR. On completion of the ISR and after clearing and re-setting the VIA interrupt enable register, the compute returns to the main program and continues from where it left off.

Note:

It should be possible to implement a similar routine using the 6809's Fast Interrupt Request Line (FIRQ). The FIRQ line may be accessed through pin 8 of the Dragon's output port; it is used on the machine to sense the presence of a cartridge.

The beauty of interrupts is that the computer does not need to concern itself with its peripheral devices until it receives an interrupt. It may therefore spend the majority of its time on some other task.

Whereas in the case of the polling routine in Program I the computer has to be always going back to look at the status of its input ports in case some action is required.

Using the non-maskable interrupt has some drawbacks for the computer is forced to cease whatever it happens to be doing and respond immediately, whereas if the FIRQ is used the computer can be programmed to complete the task in hand before it responds to the interrupt. However, of the two choices, the NMI seems to be the simpler to understand.

Using the MC6821 PIA

The pin out of the MC6821 PIA and the corresponding connections to the Dragon 32 I/O port are given in Fig. 9.

The CS0 and CS1 pins of the PIA are taken to +5 volts, while the CS2 pin is taken to P2 (Pin 36) on the Dragon I/O port.

The RS0 and RS1 pins are taken to the Dragon A0 and A1 lines respectively.

Consequently, the PIA may be addressed in the range HFF40-HFF43.

There are six locations within the PIA accessible to the MPU Data Bus. Namely, two Peripheral Registers, two Data Direction Registers and Two Central Registers. Selection of these registers is controlled by the RS0 and RS1 inputs together with bit 2 in the control register:-

Control Register Word Format for the A Port
(Format for B Port is the same)

7	6	5	4	3	2	1	0
IRQ A1	IRQ A2	CA2 Control			DDRA Access	CA1 Control	

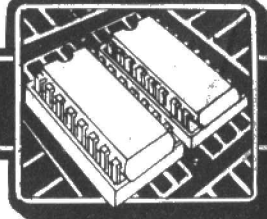
Selection of the PIA Registers

RS1	RS0	Control Register		Register Selected	Dragon Address
		CRA 2	CRA 2		
0	0	1	X	Peripheral Register A	HFF40
0	0	0	X	Data Direction Register A	HFF40
0	1	X	X	Control Register A	HFF41
1	0	X	X	Peripheral Register B	HFF42
1	0	X	0	Data Direction Register B	HFF42
1	1	X	X	Control Register B	HFF43

X = Don't care

It will be noted that the Peripheral Registers and Data Direction Registers both share the same address, which one is chosen being determined by the setting of Bit 2 in the appropriate control register.

Therefore, to use the PIA in the Output Mode -



And to use the PIA in the Input Mode—

A PORT	B PORT	COMMENT
POKE HFF41,251	POKE HFF43,251	Zeros Bit 2 of Control Register thus selecting the Data Direction Register
POKE HFF40,255	POKE HFF42,255	Sets all DDR Bits to 1 thus the corresponding lines in the Peripheral Control Register are set to Output
POKE HFF41,4	POKE HFF43,4	Sets Bit 2 of the Control Register thus selecting the appropriate Peripheral Register
POKE HFF40,DATA	POKE HFF42,DATA	Outputs Data on Peripheral Register Control lines

A PORT	B PORT	COMMENT
POKE HFF41,251	POKE HFF43,251	Zeros Bit 2 of the Control Register thus selecting the Data Direction Register
POKE HFF40,0	POKE HFF42,0	Sets all DDR Bits to 0 thus the corresponding lines in the Peripheral Control Register are set to Input
POKE HFF41,4	POKE HFF43,4	Sets Bit 2 of the Control Register thus selecting the appropriate Peripheral Register
PEEK (HFF40)	PEEK (HFF40)	Read Data Input on Peripheral Register Control Lines

Listing for Program II (Machine Code Portion)

Address	Op. Code	Source	Code	Comments
7F38	86 7E	LDA	#126	
7F3A	B7 01 09	STA	\$01 09	
7F3D	CC 7F 8B	LDD	#32651	
7F40	FD 01 0A	STD	\$01 0A	
7F43	86 7F	LDA	#127	Initialise VIA
7F45	C6 00	LDB	#0	
7F47	B7 FF 4E	STA	\$FF 4E	
7F4A	F7 FF 4C	STB	\$FF 4C	
7F4D	86 82	LDA	#130	
7F4F	B7 FF 4E	STA	\$FF 4E	
7F52	8E 06 00	LDX	#1536	
7F55	BF 30 00	STX	\$30 00	
7F58	8E 04 00	LDX	#1024	Screen
7F5B	86 80	LDA	#128	black out
7F5D	A7 80 04 00	STA,X+		
7F61	BC 30 00	CMPX	\$30 00	
7F64	26 F7	BNE	\$7F 5D	
7F66	8E 04 00	LDX	#1024	
7F69	86 05	LDA	#5	
7F6B	B7 12 00	STA	\$12 00	
7F6E	86 8F	LDA	#143	
7F70	A7 84 04 00	STA,X		
7F74	C6 FF	LDB	#255	Main Routine
7F76	5A	DECB		
7F77	26 FD	BNE	\$7F 76	
7F79	7A 12 00	DEC	\$12 00	7F7C
7F7C	26 F6	BNE	\$7F 74	
7F7E	86 80	LDA	#128	
7F80	A7 80 04 00	STA,X+		
7F84	BC 30 00	CMPX	\$30 00	
7F87	26 E5	BNE	\$7F 8E	
7F89	27 DB	BEQ	\$7F 66	
7F8B	8E 04 00	LDX	#1024	Start of ISR
7F8E	86 49	LDA	#73	
7F90	A7 80 04 00	STA,X+		
7F94	BC 30 00	CMPX	\$30 00	
7F97	26 F5	BNE	\$7F 8E	
7F99	86 10	LDA	#16	
7F9B	B7 30 41	STA	\$30 41	
7F9E	CC 30 00	LDD	#12288	
7FA1	83 00 01	SUBD	#1	
7FA4	26 FB	BNE	&7F A1	
7FA6	7A 30 41	DEC	\$30 41	
7FA9	26 F3	BNE	\$7F 9E	
7FAB	B6 FF 41	LDA	\$FF 41	
7FAE	4F	CLRA		
7FAF	86 7E	LDA	#126	
7FB1	B7 01 09	STA	\$01 09	
7FB4	CC 7F 8B	LDD	#32651	
7FB7	FD 01 0A	STD	\$01 0A	
7FBA	86 7F	LDA	#127	
7FBC	C6 00	LDB	#0	
7FBE	B7 FF 4E	STA	\$FF 4E	
7FC1	F7 FF 4C	STB	\$FF 4C	
7FC4	86 82	LDA	#130	
7FC6	B7 FF 4E	STA	\$FF 4E	
7FC9	3B	RTI		

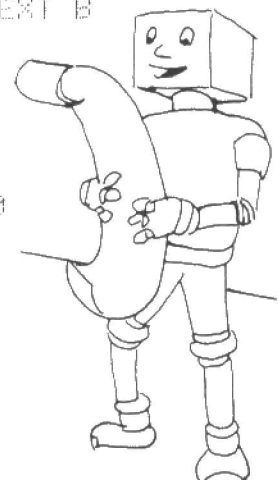
For information on the use of the control lines and interrupt inputs, the reader is referred to the MC6821 Data Sheets.

Program I

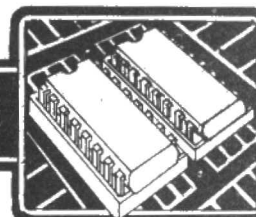
```

10 'SIMPLE I/O ROUTINE
20 CLS:PRINT@5," 'I/O' PORT TEST"
30 R=1:PRINT@35,"B PORT=IN
   A PORT=OUT"
40 POKE &HFF43,255:POKE &HFF42,0
50 FOR J=0 TO 255
60 X=PEEK(&HFF40):POKE &HFF41,X
70 IF X=100 THEN 150
80 PRINT@CR+1)*32,"CURRENT
   B VALUE=";X
90 R=R+1:IF R>=14 THEN GOSUB 130
100 FOR D=0 TO 200:NEXT D
110 NEXT J
120 END
130 FOR Z=1088 TO 1535:POKE Z,96
140 NEXT Z:R=3:RETURN
150 CLS:FOR A=0 TO 10
160 PRINT@232,"ALARMS GO OFF"
170 SOUND 60,10
180 PRINT@232,"alarms go off"
190 FOR B=0 TO 50:NEXT B
200 NEXT A
210 PRINT@387,
   "DISABLE? THEN
   PRESS ANY KEY"
220 A$=INKEY$:
   IF A$="" THEN 220
230 GOTO 20

```



Program II



```

1  *INTERRUPT DEMONSTRATION
10 CLEAR 200,32567:M=32568:DEFUSP0=32568
20 FOR I=0 TO 145
30 READ B:POKE M+I,B:NEXT I
40 DATA 134,126,183,1,9,204,127,139,253,1,10,134,127,198,0,183,255,78,247,255,76
   ,134,130,183,255,78,142,6,0,191
50 DATA 48,0,142,4,0,134,128,167,128,4,0,188,48,0,38,247,142,4,0,134,5,183,18,0,
   ,134,143,167,132,4,0
60 DATA 198,255,90,38,253,122,18,0,38,246,134,128,167,128,4,0,188,48,0,38,229,39
   ,219,142,4,0,134,73,167,128,4
70 DATA 0,188,48,0,38,245,134,16,183,48,65,204,48,0,131,0,1,38,251,122,48,65,38,
   ,243,182,255,65,79,134,126
80 DATA 103,1,9,204,127,139,253,1,10,134,127,198,0,183,255,78,247,255,76,134,130
   ,183,255,78,59
90 CLS:PRINT@6,"INTERRUPT ROUTINE"
100 PRINT@74,"USING NMI"
110 PRINT@161,"TO INITIATE NMI TAKE CA1 LINE"
120 PRINT@232,"MOMENTARILY LOW"
130 PRINT@289,"RESET TO RETURN TO BASIC MODE"
140 PRINT@355,"PRESS ANY KEY WHEN READY"
150 A$=INKEY$:IF A$=""THEN 150
160 X=USR0(0)
170 END
7000 CLS:FOR X=32568 TO 32715
7010 PRINTX;"  "PEEK(X)
7020 A$=INKEY$:IFA$=""THEN 7020
7030 NEXT X

```

SOFTWARE LIBRARY FOR SPECTRUM

Low-cost weekly hire of Games, Adventures, Utilities and Educational Programs.

We have a large selection of software in our free catalogue for Electronics & Computing readers.

- Membership only £6 for 12 months.
- Program hiring from only 80p (plus 25p p&p).
- New titles constantly being added.
- All titles with publishers permission and royalties paid.

Join today by clipping the coupon below or send for your free catalogue and see for yourself our fantastic range of software.

- ☐ YES, please send me my free catalogue and selection sheet. I enclose my £6 cheque/Postal Order.
- ☐ I enclose a 15½p stamp, please rush me your free catalogue.

NAME.....

ADDRESS.....

..... TEL.

Send to:

KERNOW SOFTWARE LIBRARY
(Dept ECM)
55 ELIOT DRIVE, ST GERMAN'S
SALTASH, CORNWALL PL12 5NL

ECM05

NEED ENCLOSING?

Now, GSC are really in the hardware business, with a series of plastic cases ideally suited to applications ranging from hand-held probes to hi-fi equipment. GSC cases are moulded in robust plastic and come with all the necessary screws, covers and, where appropriate, battery compartments, connectors and transparent panels for displays. And GSC can provide customer-specified variations for large-quantity orders. Fill in the coupon for more details.

GSC (UK) Ltd., Unit 1, Shire Hill Industrial Estate, Saffron Walden, Essex CB11 3AQ.
Telephone: (0799) 21682. Telex: 817477.

 CTP Probe Case (£4.50 Nett) 6.01	 DMC-2 Design-Mate Case (£5.75 Nett) £7.76	 CBP-1 Portable Case (£8.75 Nett) £11.21
 CTH-1 Handheld Case (£5.50 Nett) £7.47	 DMC-1 Design-Mate Case (£6.00 Nett) £8.06	 CTB-1 Bench Topper Case (£12.95 Nett) £16.04

Bold prices include P & P and 15% VAT

I enclose cheque/PO for £.....
or debit my Barclaycard, Access, American Express card

No..... Exp date.....
or Tel: (0799) 21682 with your card number and your order
will be in the post immediately.

NAME.....

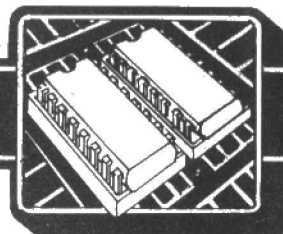
ADDRESS.....

GLOBAL SPECIALITIES CORPORATION
GSC

FREE catalogue tick box ☐

Global Specialities Corporation (UK) Limited, Dept. 34MM
Unit 1, Shire Hill Estate, Saffron Walden, Essex CB11 3AQ

TOMORROW'S TOOLS TODAY



SPECTRUM PRINTER INTERFACE

Drive a RS232 printer from normal PRINT statements by John Williams

Whilst the Sinclair ZX Printer is excellent value for money and can give good results there are occasions when the ability to use a standard printer is invaluable. Such uses include word processing, producing printouts for reports or just to give a long listing on cheap paper. This article details the design of an RS232 output port and the software to drive it from ordinary PRINT statements. The design can equally be used to communicate with other machines and to complement this an optional RS232 input is also shown.

INTRODUCTION

There are a number of methods to connect printers to computers, the two most commonly used are referred to as RS232 and Centronics. The first uses basically a single wire over which character codes are sent serially. In the Centronics system the information for each character is sent in parallel over seven wires and other wires are used to control the transfer. The two systems are quite incompatible and this article will concentrate solely on the RS232 serial interface.

The term RS232 defines a serial interface system that has a wide range of uses beyond that for just printers. These include Modems VDUs and data links. Whilst the voltage levels, connector types and pin allocations are specified a number of options are provided to meet the needs of different instruments. To ensure compatibility with all devices that may be encountered a computer must be provided with a measure of versatility. The greatest variations will be found in the code standards that are used, fortunately there are many integrated circuits available that cater for the full range of codes likely to be needed. It is one of these that enables this design to be achieved relatively simply.

Fig. 1 RS232 Connector

Pin Number	Relative to Printer	
	Description	Signal Direction
1	Protective ground	
2	Transmitted data	Output
3	Received data	Input
4	Request to send	Output
5	Clear to send	Input
6	Data set ready	Input
7	Signal ground	
8	Carrier detect	Input

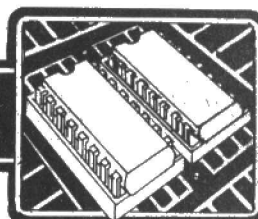
Fig. 2

Standard Baud Rates
50
100
300
600
1200
2400
4800
9600

RS232 STANDARDS

Interconnections in a RS232 system are made via a 25 way "D" connector with the socket at the computer end. The only pins normally used are shown in Fig. 1 along with their functional descriptions as seen by the printer. When reading them it must be realised that what is received data to the printer is transmitted data at the computer otherwise confusion can arise. The actual character data is sent to the printer on pin 3, this will be in serial form using the

ASCII code which requires seven bits to define each character. The frequency of transmission of the separate bits is known as the Baud rate. Fig. 2 shows the rates commonly used, the actual character transmission rates are about 1/10th the Baud rate. The manner in which the code is sent is illustrated in Fig. 3. The first bit transmitted is always a start bit of the polarity shown, after this the character code is sent and it may be followed by a parity bit. The parity may be odd or even depending on the transmission standard used and its purpose is



to enable errors to be detected by making the total number of logic 1s odd or even. There is always one and sometimes two stop bits at the end of each character. The line is then ready to pass the next character beginning with its start bit.

Handshake is often employed between a computer and a printer. This enables the printer to tell the computer to stop sending more data if it has as much as it can currently cope with. This may occur for instance after a carriage return has been sent; many printers take longer to perform this function than a normal character print. The handshake stops further characters being sent which would otherwise be lost while the carriage return is under way. A system can either have hardware handshake, software handshake or none. If none is used the Baud rate must be set slowly enough so that the information can be printed as fast as it is received.

Hardware handshake uses pin 4 on the connector entitled Request To Send. It enables transmission when logic high and disables it when low. This is the only form of handshake provided in this design although it is possible for the user to write his own software routine if required. Software handshake is the system where certain codes are sent in both directions between computer and printer to start and stop the transmission of character codes. To use this method requires that the receive as well as transmission paths are constructed.

For data communication into the computer pins 2 and 5 are used. Pin 2 carries the data input and pin 5 carries the handshake output that tells the sending end that it can start to send. It will be seen that pins 2 & 5 are complementary to pins 3 & 4. If two computers are to communicate on the RS232 line then a special crossover cable is required with pin 2 at one end going to pin 3 at the other end and vice-versa, similarly for pins 4 & 5. This is shown in Fig. 4.

A problem sometimes encountered when connecting up a printer is the presence or absence of auto-linefeed facility. Line feed is the instruction to move down one line in the printing. Some computers like the Spectrum end their lines with only a carriage return (code OD), the printer is required to assume a line feed and go to the next line; this is referred to as auto-linefeed. Some printers however require a separate line

Fig. 3 Serial Transmission Format

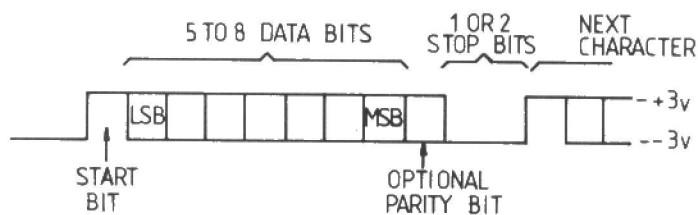


Fig. 4 Crossover Cable for Interconnecting Computers



feed (code OA) otherwise all the printing will be on one line. This design provides for the addition of an optical line feed after every carriage return by POKING a specified memory location during the initialising procedure.

Another common problem is differing character sets. The character set defines the shapes of the characters that are displayed or printed in response to character codes. There is no universally used standard; although most machines conform loosely to ASCII and will correctly respond to 0 to 9, A to Z and arithmetic symbols some of the other characters often vary. For instance only English language printers use £ and not many except the Spectrum use ©. There is little the user can do except remember the true meaning of those different symbols his printer gives. This problem does not arise with the ZX Printer because its character set is held in the computer and the printer is told what to do dot by dot.

OPERATIONAL DESCRIPTION

The interface design is based on an Intersil universal asynchronous receiver/transmitter (UART) integrated circuit that performs all the work of coding data into serial form. The only extra circuitry needed is a clock oscillator, simple

address decoder and RS232 line buffers. The circuit is shown in Fig. 5 and for printer use only the part above the dotted line is required. The operation of the circuit is briefly explained below. The address decoder IC3a will send a write pulse to the UART whenever an OUT instruction is performed to port number 65407. This is distinguished from the memory address of the same value by the IORQ line being active low. Although the circuit only requires the port number to have address A7 low the other address lines must be high to avoid operating other devices within the Spectrum. The character code will be latched off the data bus by the UART and stored in its transmitter buffer register from where it will automatically be sent along with start, stop and parity bits on the serial output pin 22. Because the serial transmission is much slower than the normal working speed of the computer there is likely to be data available more frequently than the UART can send it. To avoid losing data the UART will cause the WAIT line to be pulled to active low whenever its transmitter buffer register is full indicated by the TRBE pin 22 being low. If the handshake RTS is low and a further character is ready to be written to the UART then IC3b will go low thus causing a WAIT until RTS goes high. The RS232 input buffer IC7 will operate

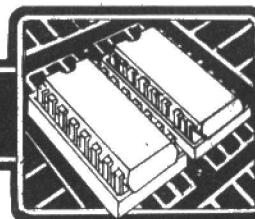
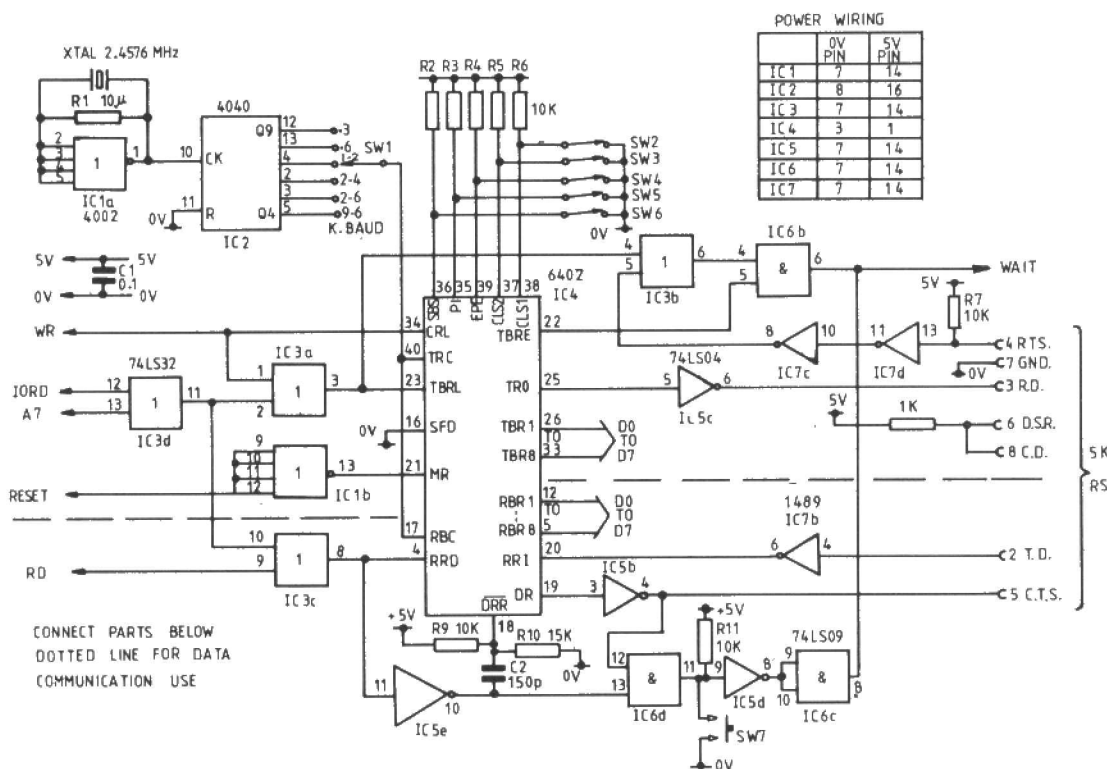


Fig. 5 Circuit Diagram



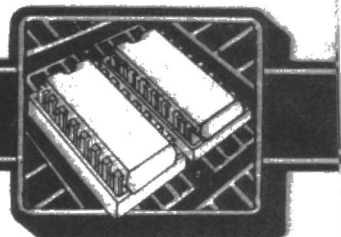
Parts List for Fig. 5

IC1 4002 CMOS
IC2 4040 CMOS
IC3 74LS32
IC4 IM6402
IC5 74LS04
IC6 74LS09
IC7 MC1489
C1 0.1 μ F
C2 * 150 pF
R1 10 M Ω
R2 to R7 10 K Ω
R8 1 K Ω
R9, R11 * 10 K Ω
R10 * 15 K Ω
SW1 1 pole 6 way DIL switch
SW2 to SW6 SPST DIL switch
SW7 * Miniature push button
normally open
XTAL 2.4576 MHz
SKT1 28 way double sided edge
connector
SKT2 25 way "D" connector

* Required for data input only.

Fig. 6 Transmission Standard Selection

Character length (bits)	5	6	7	8
SW2	ON	OFF	ON	OFF
SW3	ON	ON	OFF	OFF
Parity	ODD	EVEN	NONE	NONE
SW4	ON	OFF	ON	OFF
SW5	ON	ON	OFF	OFF
No. of Stop bits	1	2		
SW6	ON	OFF		



on TTL inputs but will also accept levels up to ± 15 volts without damage. The serial data is transmitted to the RS232 line via IC5c. This is a normal TTL gate and its input does not fully conform to the RS232 voltage levels in respect of the logic low level. This should be -3 volts however the driver gives about 0 volts, it will be found in practice that nearly all RS232 printers will work correctly with the TTL levels provided. This short cut in the design was made because of the difficulty in obtaining a suitable negative power rail from the Spectrum.

The Baud rate is selected by switch SW1 which connects the required frequency from the counter to the clock inputs of the UART. The clock frequency is 16 times the Baud rate. The switches SW2 to SW6 select the transmission standard that is used by the UART. Fig. 6 shows the combination available. If the required standard for the printer is not known then setting all the switches to OFF will often suffice.

The part of the circuit below the dotted line is only needed if data communication into the computer is required. The design has been extended to include this facility since it uses no additional ICs and requires only 3 resistors, a capacitor and a push button extra. It is controlled from an IN instruction also from port number 65407. When this is done the WAIT line will go low to stop further processing until the UART indicates on its data received pin 19 that it has received a code on the serial input. The code will then be passed onto the data bus. The push button SW7 is provided as a fallback to allow the user to abort the IN operation if no data is available.

SOFTWARE

In simple applications the output port can be driven one character at a time from BASIC. The command OUT 65407, X will cause the character whose ASCII code is X to be printed. Whilst this allows full control of the printer it is far from a convenient way of printing tables, text or listings. It would be much better to be able to use normal PRINT statements. Fortunately the Spectrum is very versatile in its input/output ability and is designed to have the facility to use other than the standard channels of Screen, Keyboard or ZX Printer.

Fig. 7

```
FE00 3E 00 18 02 3E 02 32 B0 5C 01 16 FE ED 43 C5 5C
FE10 3E 48 32 81 5C C9 FD CB 76 46 0E 7F 20 30 FD CB
FE20 76 56 20 0C FE 06 28 16 FE 16 28 0D FE 17 20 4C
FE30 FD CB 76 C6 FD CB 76 96 C9 FD CB 76 D6 C9 FD 7E
FE40 77 E6 0F C8 3E 20 CD 7D 1E FD 34 77 18 F0 FD CB
FE50 76 86 FD BE 77 38 07 FD 96 77 47 04 18 19 47 04
FE60 FD 36 77 FF 3E 0D CD 7D 1E FL CB 76 4E 28 05 3E
FE70 04 CD 7D 1E FD 34 77 3E 20 10 F6 C9 FE A5 30 48
FE80 FE 0D 28 32 FE 20 D8 FE 90 38 02 D6 2F FE 80 38
FE90 02 D6 1F FD 34 77 47 3A 81 5C FD BE 77 30 14 3E
FEA0 0D CD 7D 1E FD 36 77 01 FD CB 76 4E 18 05 3E 0A
FEB0 CD 7D 1E 78 18 0F FD 36 77 00 FD CB 76 4E 28 05
FEC0 CD 7D 1E 3E 04 C3 7D 1E 21 95 00 D6 A4 47 CD 30
FED0 03 23 E6 80 28 F8 05 20 F5 3E 2C CD 7D 1E FD 34
FEE0 77 FD 34 77 3A 81 5C FD BE 77 30 14 3E 0D CD 7D
FEF0 1E FD 36 77 00 FD CB 76 4E 28 05 3E 0A CD 7D 1E
FFF0 CD 30 03 23 CB 17 38 07 CB 3F CD 7D 1E 18 D2 CB
FF10 3F CD 7D 1E FD 34 77 3E 20 C3 7D 1E CC 00 00 00
```

Whenever a PRINT or INPUT statement is executed the microprocessor will read from memory the address of the routine that is to be used. These addresses are held in RAM between 5CB6 and 5CC9 in the area called Channel Information. These RAM locations are set up at power up to point to the addresses of the appropriate ROM routines. All that has to be done is to POKE the printer output routine pointer to be the address of user software. Every time a LPRINT, LLIST or PRINT # 3 is done the output will then automatically go to the RS232. The addresses to POKE are 23749 and 23750 (5CC5 & 5CC6 hex) which are to contain the 16 bit user routine address in the usual Z80 manner of low byte first.

In its elemental form the output routine has to take the character code which is in the A register and OUT it to port address 7F. This works alright for alphanumerics but will not spell out keywords or respond to control functions such as TAB. The programme given does both these and will automatically insert a carriage return to give rollover if a line exceeds a predefined length. The line length is stored as a number in memory location 23681 (5C81) and must be POKED prior to running. Additionally the programme monitors memory 23728 (5CB0) and if

Fig. 8 Initialisation Routine Mnemonics

Addr	Hex	Data	Op	Operands
Initialisation Routine				
FE00	3E00	LD	A, 00h	
FE02	1802	JR	+02h	
FE04	3E02	LD	A, 02h	
FE06	32B05C	LD	(5CB0h), A	
FE09	0118FE	LD	BC, Output R	
FE0C	ED43C55C	LD	(5CC5h), BC	
FE10	3E48	LD	A, Line length	
FE12	32B15C	LD	(5CB1h), A	
FE15	C9	RET		
Output Routine				
FE18	FDCB7656	BIT	0, (IY+76h)	

this has been POKED to be 2 a line feed will be added after every carriage return. To save having to do these POKES a machine code programme to do the initialisation is given and can be saved along with the output routine itself. This is called once with a PRINT USER instruction.

The output routine has been made relocatable and can be put anywhere in memory. Only the initialising routine has to be altered to point to the correct starting address. To show which two addresses to change the initialising routine is shown additionally in annotated mnemonic form in Fig. 8. The line length number can also be changed if required; a loader programme is shown

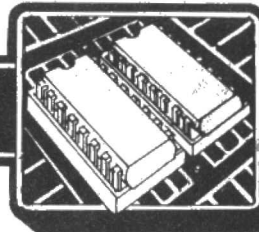


Fig. 9 Loader Programme

```

5 CLEAR 65000
10 LET a=65024
15 RESTORE
20 READ c$
30 LET c=CODE c$-48-(7*(c$(1)>
"@"))
40 LET d=CODE c$(2)-48-(7*(c$(
2)>"@"))
50 POKE a,16*c+d
60 LET a=a+1
70 IF LEN c#=2 THEN GO TO 20
80 LET c#=c$(3 TO )
90 IF c$(1)<>" " THEN GO TO 30
100 LET c#=c$(2 TO ): GO TO 90
180 REM      Must use CAPITALS
           in Data statements.
200 DATA "3E 00 18 02 3E 02 32
80 5C 01 16 FE ED 43 C5 5C"
210 DATA "3E 48 32 81 5C C9 FD
CB 76 46 0E 7F 20 30 FD CB"
900 REM      Continue for rest
           of code

```

in Fig. 9 which will put the code into the top of the RAM of the 48K Spectrum above 65024. Once this has been run the code can be saved on tape, its length is 284 bytes. The statement CLEAR 65000 should be executed before re-loading the code. For the 16K Spectrum subtract 32768 from all quoted numbers and alter the 12th code byte from FE to 7E. If the EPROM board previously described in this magazine has been built then this is the best place for the code as it will always be instantly available. It can be put in the spare part of the EPROM above 3C40. The initialisation routine when loaded is run by PRINT USER 65024 if the option line feed is not required, or PRINT USER 65028 if it is. After this no further action is required and all printing that would normally go to the ZX Printer will now go to the RS232 port.

No software for an input routine has not been produced however the BASIC instruction IN 65407 can be used to read in individual character codes.

USED EXTENSIVELY THROUGHOUT THE ELECTRONICS INDUSTRY AND NOW AVAILABLE FOR THE FIRST TIME TO THE AMBITIOUS CONSTRUCTOR, THE R50 DIY COMPUTER FROM RADE SYSTEMS. HERE'S WHAT YOU GET...

● Z80A BASED PROCESSOR ● FULL 64K OF DYNAMIC RAM ● DUAL SERIAL PORTS ● PARALLEL PRINTER PORT
● UNLIMITED EXPANSION VIA TWO EXPANSION CONNECTORS ● FULL 4MHZ CPU ● FULL CPM SUPPORT VIA ADD-ON FLOPPY OR HARD DISC OPTION BOARDS ● USER MANUAL

You can make it.

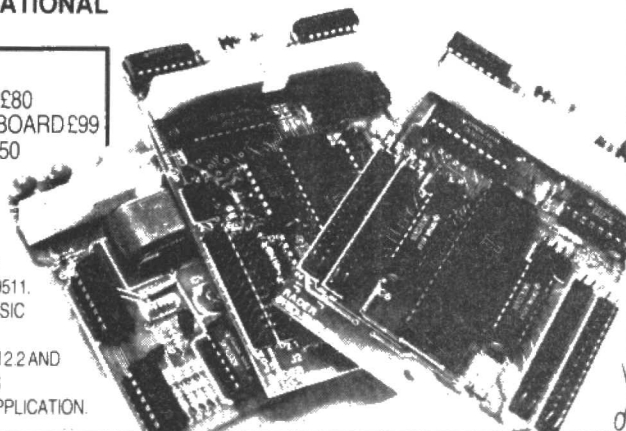
DOZENS OF APPLICATIONS INCLUDE: ● AUTOMATION DEVICES
● CONTROL APPLICATIONS ● COMMUNICATIONS EQUIPMENT
● BUSINESS MACHINES ● WORD PROCESSORS ● GRAPHICS DISPLAYS
● SCIENTIFIC AND EDUCATIONAL

DEALER ENQUIRIES WELCOME

OPTION BOARDS INCLUDE:

- FLOPPY DISK CONTROLLER £80
 - SPRITE COLOUR GRAPHICS BOARD £99
 - R50 TECHNICAL MANUAL £6.50
- PRICES INCLUDE P+P

ALSO AVAILABLE PIO, SIO, IEEE 488 PORT, REAL TIME CLOCK, PROTOTYPING BOARD, MATHS PROCESSOR USES AMD9511, A TO D CONVERTOR, STEREO SOUND MUSIC BOARD, 192K RAM BOARD, CASSETTE INTERFACE, HARD DISC INTERFACE, CPM 2.2 AND TURBODOS AVAILABLE. INTERFACE KITS FOR ALL LEADING MICROS, PRICES ON APPLICATION.



ONLY
£215
PRICE INCLUDES VAT, PACKING + POSTAGE

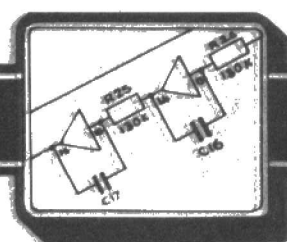
PLEASE MAKE CHEQUES/POSTAL ORDERS PAYABLE TO RADE SYSTEMS LTD.

R50 DIY SINGLE BOARD COMPUTER £215
FLOPPY DISC CONTROLLER £80
SPRITE COLOUR GRAPHICS BOARD £99
R50 TECHNICAL MANUAL £6.50
MORE INFORMATION...
AMOUNT ENCLOSED £

NAME
ADDRESS
POST CODE

RADE SYSTEMS LTD.
290A HIGH ROAD,
WILLESDEN,
LONDON
NW10 2EU

RADE
DIY computer



RULES FOR DIALOGUE PROGRAMMING

by Philip Barker

Writing programs that are able to sustain acceptable real-time dialogue with computer users is often not an easy task. Indeed, the problems involved are often significantly more difficult than those encountered in conventional programming tasks. When writing programs of this type one of the most important factors to bear in mind is 'user-psychology'. It is imperative that the dialogue designer and programmer has some understanding of how a potential user is likely to react when confronted with the various messages that a program is likely to produce. The extensive work on the psychology of communication by Miller (Mil63, Mil67) and others is thus of considerable value to those who construct interactive software. More recently, over the last few years a considerable amount of practical experience has been gained in this area. Gaines, for example has formulated a set of seventeen 'dialogue programming rules' that provide many useful guidelines for the design of interactive dialogue driven software (Gai81). Embodied on these rules is a variety of generally useful principles. Some of these are briefly outlined below.

Perhaps one of the most important things to realise is that users vary in their capability and sophistication. In view of this, software should be able to dynamically adapt to the level of the user. Because users will often make mistakes it should be possible for them to gracefully recover from these. Some form of back-tracking facility is therefore an important service to be provided by interactive software. Humans have a limited bandwidth for the intake of unfamiliar information. Hence, it is important to avoid providing the user

with too much information at once. Frames for its display should therefore be made as aesthetically pleasing as possible – well designed and, if possible, using colour to best advantage. Related to this issue is the design of messages; these should be as comprehensible as is possible. Primarily, computer systems must be designed to serve people – rather than the other way around. In view of this, the dialogue designer must let the user 'drive' the software at the rate he/she wishes to proceed. Of course, during the dialogue it should always be a simple matter for the user to find out where he/she is within a particular system or sub-system. Finally, the designer should provide a facile mechanism to enable the user to regain control of a situation if the need ever arises. There are many books devoted to the principles of dialogue design (Kid82, Mea70). Further details on the above topics will be found in these.

Having briefly outlined: (1) the necessary requirements of a dialogue programming language, and, (2) some of the rules that should be observed in programming human-computer dialogues we next outline the PILOT system and present some simple examples of programs written in this language. We conclude with an assessment of the language in terms of the basic criteria outlined above.

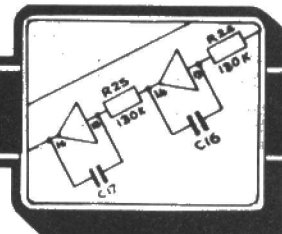
THE PILOT SYSTEM

PILOT is an acronym for Programmed Inquiry, Learning Or Teaching. The original system was formulated and

implemented in 1973 by John Starkweather at the University of California. Over the last ten years a variety of PILOT implementations have sprung into existence (Mun80). Common PILOT and MTC PILOT are two popular examples. The language is now available on a wide variety of micro-computers (PET, APPLE, TANDY, ATARI, etc.) and minicomputer systems (PRI81). Conceptually, the language is very simple since most implementations contain fewer than a dozen statement types. In view of this, and the simplistic structure of the individual statements it is an extremely easy language to learn. A summary of the general statement structure and the different types of statement is given in figure 4.

Each statement can contain up to four parts: a label, an operation code (or command), a modifier (that influences the effect of the command), and, an operand. The operation code specifies some fundamental operation that the computer is to perform; the operand then provides the information that is necessary to perform this operation. Labels are used to identify points in a PILOT program that may be used as the target of jump-statements (J) – these are analogous to GOTO's in BASIC. A label may stand on a line of its own or may occur in the leading position of a more complex statement. As can be seen from the examples given in figure 4A, the label field of a command always commences with an asterisk.

The commands used in PILOT each consist of a single letter. The allowed letters are listed in figure 4B along with a description of their function. Information output is achieved via the type command T. All symbols following the colon (:) are



are displayed on the VDU screen — unless they represent special cursor control characters. In example 2 (see figure 4A) the word 'HELLO' would appear on the user's CRT display. Similarly, example 3 would cause 'GOODMORNING' to appear while the code in the fourth example would produce this message only if the value of the numeric variable #Z was greater than zero. Because the T operation is used so frequently there are certain situations where it may be omitted. Thus, when several lines are to be output only the first of these have to include the T op-code; subsequent lines may commence with just a colon. This arrangement is illustrated in the simple program shown in figure 4D.

Information input from the user is achieved via the Accept command (A). If an operand is specified then the user's response is assigned to the one (or more) variables that it contains. Otherwise, the response is stored in a special system variable for subsequent analysis. One way of performing response analysis is via the match command (M). This can be used to perform a 'sliding window' string comparison of each of its operands (in turn) against the user's response string. A switch is then set to indicate whether the match was successful or not. An example of the use of this command is given in figure 4D. Here a student is presented with a simple question. After accepting an answer, the response is matched to see if the value is '9' or 'nine'.

Should this be the case a jump is made to that part of the program bearing the label ***CORRECT**. The conditional effect of the jump statement is achieved by the Y modifier that it contains. This specifies (YES) the jump is to be taken provided the previous match was successful. In a similar fashion the N modifier could be used to force a jump if a match was unsuccessful. Other modifiers (see figure 4C) may be employed in order to specify alternative types of conditional statement execution.

When a match command is used it is possible to specify (1) a list of options that will produce a successful match, and, (2) whether exact or partial match-

FIGURE 4 AN OVERVIEW OF THE PILOT LANGUAGE

(A) Statement Structure



EXAMPLES:

- ```
(1) *START
(2) T: HELLO
(3) *START1 T: GOODMORNING
(4) *START2 T(#Z>0): GOODMORNING
```

### (B) Types of Statement

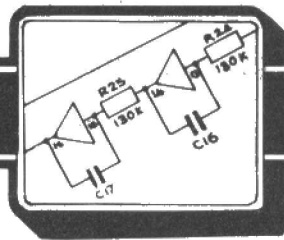
| Command Symbol | Function                            | Example              |
|----------------|-------------------------------------|----------------------|
| T              | Type a message                      | T: HELLO             |
| :              | Type a continuation line            | : What is your name? |
| A              | Accept - Information Input          | A:                   |
| M              | Match - String matching             | M: nine!%9%          |
| J              | Jump - branch statement             | J: *Q36R             |
| C              | Compute - performs calculations     | C: #A=27+36          |
| D              | Dimension - used to declare arrays  | D: SQUES(10,10)      |
| U              | Use - subroutine invocation         | U: *PICTURE          |
| E              | End - of subroutine or program      | E:                   |
| R              | Remark - used to annotate a program | R: THIS IS A COMMENT |
| V              | Output a VDU footer                 | V: Press any key     |
| P              | Page formatter                      | P:                   |
| X              | Chain in and execute next program   | X: ELEC.PART2        |

### (C) Command Modifiers

|        |                                              |
|--------|----------------------------------------------|
| Y      | Yes - jump is to be taken if positive match  |
| N      | No - jump is to be taken if no match         |
| <cond> | condition modifier                           |
| C      | Condition continuation                       |
| E      | Error                                        |
| H      | Cursor co-trol on type and accept statements |
| J      | Automatic jump on no match                   |

#### (D) A Simple PILOT Program

```
*BEGIN
T: What is the answer to the following calculation
 3+6 = ??
A:
M: %9!%nine%
JY: *CORRECT
M: %18!%eighteen%
JY: *PRODUCT
T: No. You are wrong.
: Have another go.
J: *BEGIN
*PRODUCT
T: No - you've given me the product.
: Have another go.
J: *BEGIN
*CORRECT
T: Well done.
: You are correct.
```



ing is required. The latter type of matching is most often used when text string responses from the user are being analysed. In this situation it is possible to intersperse certain special characters in the operand options of the M command. These influence the way in which pattern matching proceeds. For example:

- % matches any number of blanks
- \* matches any one character
- & matches any sequence of characters

Their use is illustrated in the examples shown in figure 5.

Diagram 5A illustrates the use of the compute statement (C) and two PILOT built-in functions (TIME and INS). The compute statement has to be used whenever it is required to perform some form of calculation. Virtually, any BASIC-like assignation can be used as the operand of this command. In example A, the first compute operation is used to assign the time of day (in the form HH:MM:SS) to the string variable \$Y. The following statements perform simple pattern matching via the in-string function (called INS) in order to determine if the time is before midday. The INS function is used to effect substring searching. Its calling mechanism is simply INS(P,S,T) in which P is the substring to be matched, T is the target string to be searched and S is the position within T at which the search is to commence. If a successful match is made the value returned is the position within T at which an instance of P commences. An unsuccessful match causes a zero value to be returned. It is easy to see that the code presented in figure 5A will print out either "Good Morning" or "Good Afternoon" depending upon the time at which it is invoked.

Another important application of pattern matching arises in situations where students may mis-spell words or where there are alternative spellings. The latter is particularly important if both American and English spellings are to be catered for. PILOT code suitable for handling the variants of the string 'analogue to digital convertor' is depicted in figure 5B. The match command used in this example would give a positive result with any of the following student responses:

## (A) Use of Pattern Matching Functions

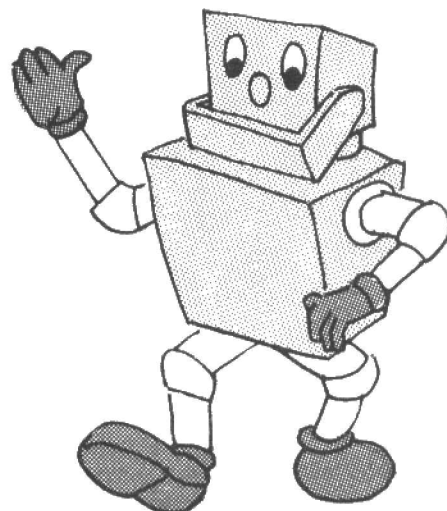
```
R: Ask the computer the time.
C: SY=TIME
R: Now see if its before noon.
C: #Z1 = INS("0*: **: **: ",1,$Y)
C: #Z2 = INS("10*: **: **: ",1,$Y)
C: #Z3 = INS("11*: **: **: ",1,$Y)
C: #Z = #Z1+#Z2+#Z3
T(#Z>0): Good Morning
T(#Z=0): Good Afternoon
: Today we shall study Computing
: Tomorrow we shall do Electronics
T: What is your name?
A: SNAME
```

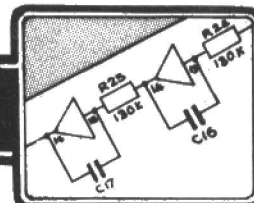
## (B) Use of the Match Command

```
*QUES9
T: What is the name of the device normally used for changing
: real world signals into digital form?
A:
M: ADC!ADC&!analog&to%digital%convert*!analog&to%digital%convert*&
TY: Well done.
JY: *QUES10
T: No. You haven't got it.
J: *REVISION
*QUES10
T: What type of IC would you use to combine signals from
: different sources into a single output line?
A:
J: *END
*REVISION
T: Let's do some revision.
*END
R: End of Program
```

ADC  
ADC chip  
ADC IC  
analogue to digital converter  
analog to digital convertor chip  
etc.

Another useful feature of PILOT is the variety of built-in functions it contains. Two of these have already been introduced: TIME and INS. Many other useful ones are available. Typical examples of these include RND for generating random numbers; DATE which yields the date; and, CHR for converting ASCII codes to their corresponding characters. CHR is extremely useful for sending control information to any ancillary devices that are connected to the student's primary input device (a VDU or micro). In addition, there is a range of functions for string handling (LEN, LEFT, MID, RIGHT) analogous to those available in BASIC, and other programming languages.





# THE BBC MICRO AND FOURIER SYNTHESIS

Paul Beverley  
Norwich City College

If you are trying to teach about sound and in particular about waveforms, you want to show that any periodic waveform can be made up of a number of sine waves of different frequencies. To the basic fundamental frequency you add different amounts of different harmonics, and by adding these together you get the particular shape of waveform that you are looking for. This is what is known as "Fourier Synthesis". The reverse of this, "Fourier Analysis", is where you start with a waveform and try to find out how much of each of the harmonics it contains.

You can buy a piece of electronic equipment, known as a spectrum analyser, which will perform fourier analysis on a waveform, but it is extremely expensive. You can also buy units which allow you to synthesise waveforms by combining together different amounts of different harmonics and also with different phase shifts, but these are also very costly.

With the BBC microcomputer we can do some simple fourier synthesis which can be used fairly effectively to put across the basic idea of how waveforms are built up. In this article I shall be looking at two different programs. The first uses mathematical calculations to work out the proportions of different harmonics needed to produce square waves, triangular waves and sawtooth waves. It adds up all the harmonics and at each stage displays the resultant waveform on the screen.

The second program takes a different approach. It allows the user to specify the proportions of the 3rd, 5th and 7th harmonics including allowing negative values i.e. with a phase shift of 180 degrees. It then adds up the fundamental, plus the three harmonics, displays them on the screen and also actually synthesizes them using a digital to analogue converter which can be placed on either the user port or the printer port. As written, the program assumes that you are using the user port, so to change to the printer port you would simply have to change line 140. You can then take the output from the converter and feed it into an amplifier and loud speaker so that you can hear the resulting waveform.

## Square, Triangular, or Sawtooth?

Let us look in some detail at the two programs. The first one allows you to generate a square wave, a triangular wave or a sawtooth wave. First of all it draws two cycles of the basic sine wave, and then it goes through each of the succeeding harmonics calculating the proportions required, according to a different mathematical formula for each waveform, and drawing to scale the harmonic on the screen. It then adds it on to the original waveform and draws the resultant waveform.

As this program runs, you will see that the waveform gets

closer and closer towards the theoretical square, triangular or sawtooth. In the case of the triangular, since the basic shape is similar to that of a sine wave, it does not take very many harmonics to reach the required shape. The size of each harmonic, whose value is printed on the screen as a percentage, is seen to become very small very rapidly.

In the case of the square wave, it takes somewhat longer because of the sharp corners which have to be generated. These sharp corners represent relatively large amounts of the higher harmonics. What you tend to get is some degree of overshoot before it settles down to the flat top of the square wave and this is repeated of course at each of the corners of the square wave. The sawtooth is the worst of all to generate because it is not symmetrical. It uses not only the 3rd, 5th, 7th and 9th harmonics etc., but also the even harmonics, 2nd, 4th, 6th, 8th etc. in order to get the necessary asymmetry.

The way in which the program runs is that after each curve has been generated there is a two second delay before moving on to the next. If you do not want to delay then you can press the space bar and it moves on immediately. It is possible to halt the program at any stage in order to look at the waveform and perhaps talk about it to the class if you are teaching. To do this you press any key other than the space bar and it will halt, and then to start it again you press the space bar. If you have finished with that particular waveform and want to go to another, you press the escape key and it automatically re-runs the program from the beginning.

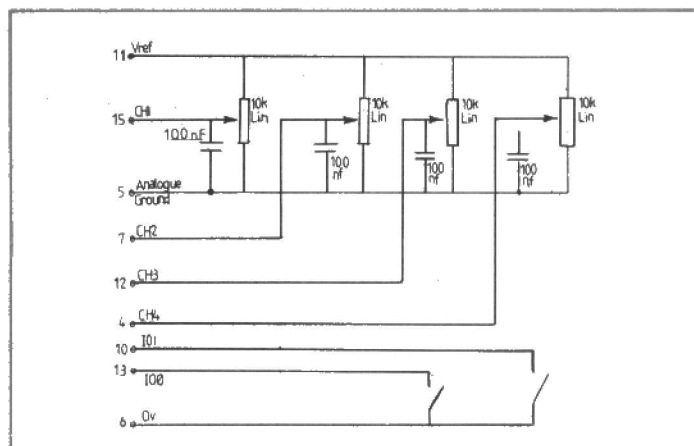
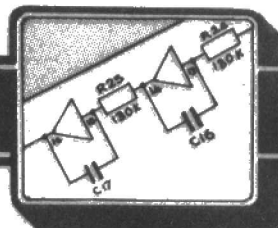


Figure 1  
Connections to the 15 way D type connector



## Do-It-Yourself Waveform Generation

The second program allows you input the proportions of the 3rd, 5th and 7th harmonics which are to be added to the fundamental. Rather than the user entering the different values from the keyboard, the idea is to use the analogue to digital converters to sense the positions of four potentiometers. Figure 1 shows how these potentiometers are wired on to the 15-way D-type connector on the back of the computer. The program makes use of the two fire buttons which are also on the 15-way connector, and these are also shown in figure 1. One of them is used to allow the user to indicate that he wants the output to be synthesized, and the other is used to stop the synthesis.

The actual synthesis is done by the short machine code program at lines 190 to 280, which, as well as outputting the data to the digital to analogue converter, also checks to see if

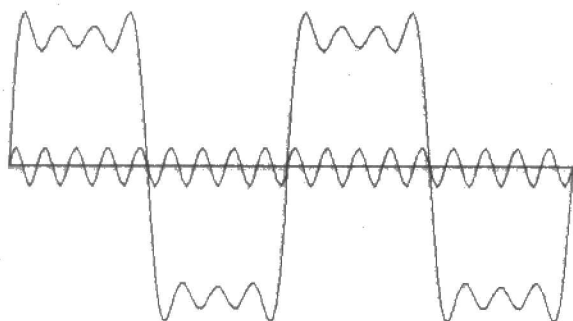
the button is pressed. The four potentiometers then are used to set amplitudes of each of the fundamental and the 3rd, 5th and 7th harmonics.

This idea of using the potentiometers and the fire buttons as a means of inputting variables into some sort of scientific simulation is a very powerful one, and has a number of different possible applications. For example, one of the most obvious examples of a useful simulation in A level physics is the Millikans experiment. If this is done in such a way that the user has to input the voltage as a number from the keyboard, it is far less realistic than actually twiddling a knob to vary the voltage in the simulation.

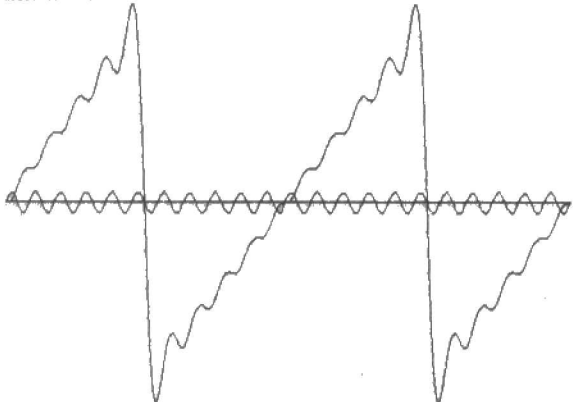
Once again, there is no suggestion that these programs are highly polished and professional. They simply show the sorts of things that can be done with the B.B.C. microcomputer. It is up to you to tailor them to your own requirements.

### Print dumps of Program 1

Harmonic Number 9  
Amplitude 12.111111 x



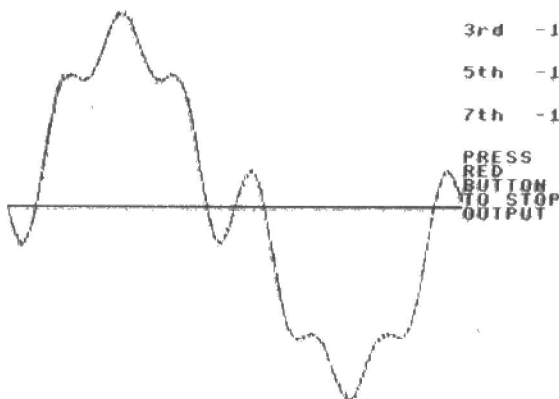
Harmonic Number 11  
Amplitude 9.999999 x



### Print dumps of Program 2

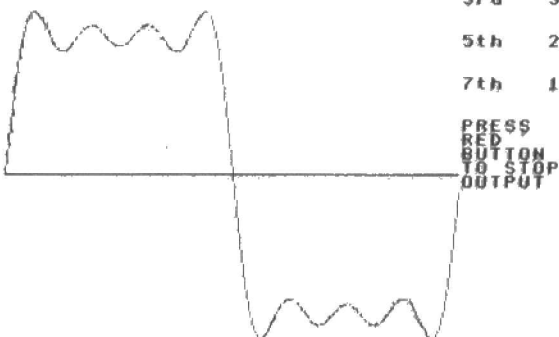
1st 88  
3rd -16  
5th -13  
7th -18

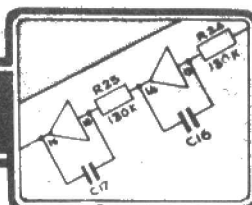
PRESS  
RED  
BUTTON  
TO STOP  
OUTPUT



1st 100  
3rd 33  
5th 20  
7th 14

PRESS  
RED  
BUTTON  
TO STOP  
OUTPUT





## Program 1

```

10 ON ERROR GOTO 740
20 MODE0
30 VDU19;4;0;
40 PRINT"1 = SQUARE""2 = TRIANGULAR""3 = SAWTOOTH"
50 PRINT"0 = FINISH"
60 INPUT"WHICH CURVE",CX
70 IFCX<1 CLS:END
80 IFCX>3 RUN
90
100 AX=430-CX*55
110 VDU23;10,32,0;0;0;
120 W=PI/320
130 DZ=1
140 SZ=4
150 NZ=1
160 VZ=636
170 DIM Y(158),RX(158)
180 CLS
190
200 REPEAT
210 VDU29,0;500;
220 PROCtitle
230
240 FORQZ=0TO158
250 XZ=4*QZ
260 comp=AX*SIN(NZ*SZ*QZ*W)/DZ
270 RX(QZ)=comp
280 DRAWXZ,comp
290 Y(QZ)=Y(QZ)+comp
300 NEXT
310
320 PROCdraw
330 PROCwait
340 CLS
350 PROCtitle
360
370 FORQZ=0TO158
380 XZ=4*QZ
390 DRAWXZ,Y(QZ)
400 RX(QZ)=Y(QZ)
410 NEXT
420 PROCdraw
430
440 IF CX=3 NZ=NZ+1 ELSE NZ=NZ+2
450 IF CX=2 DZ=NZ*NZ ELSE DZ=NZ
460 IFCX>1 SZ=-SZ
470 PROCwait
480 UNTILO
490 END
500
510 DEFPROCdraw
520 FORQZ=0TO158
530 XZ=QZ*4+VZ
540 DRAWXZ,RZ(QZ)
550 NEXT
560 DRAW1279,0
570 ENDPROC
580
590 DEFPROCtitle
600 MOVE0,0
610 DRAW1279,0
620 MOVE0,0
630 PRINTCHR$(30);"Harmonic Number ";NZ
640 amp=SZ*100/DZ/4
650 PRINT"Amplitude ";amp;" % ";
660 IFamp#8=INT(amp#8) PRINT" "
670 ENDPROC
680
690 DEFPROCwait
700 J%=INKEY$(200)
710 IF J%>" " J%=GET$
720 ENDPROC
730
740 IF ERR=17 THEN RUN
750 REPORT

```

## Program 2

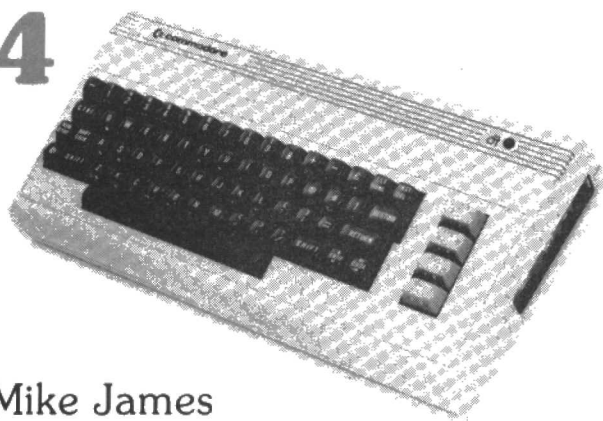
```

10 MODE4
20 PROCinitialise
30 REPEAT
40 PROCgetvalues
50 PROCplot
60 CALLstart
70 UNTILO
80
90 DEFPROCinitialise
100 QZ=4
110 VDU19;4;0;
120 VDU23;10,32,0;0;0;
130 SZ=325
140 dac=&FE60
150 DIMcomp(3,255)
160 DIM dataZ 255, PZ 30
170 start=FZ
180 OPT 0
190 SEI
200 .output
210 LDA dataZ,X
220 STA dac
230 INX
240 LDA #16
250 BIT &FE40
260 BNE output
270 CLI
280 RTS
290 J
300 k0=PI/128
310 k1=k0*3
320 k2=k0*5
330 k3=k0*7
340 FZ=100
350 H3Z=0
360 H5Z=0
370 H7Z=0
380 FOR NZ=0 TO 255
390 VDU30
400 PRINTNZ
410 comp(0,NZ)=SIN(k0*NZ)
420 comp(1,NZ)=SIN(k1*NZ)
430 comp(2,NZ)=SIN(k2*NZ)
440 comp(3,NZ)=SIN(k3*NZ)
450 NEXT
460 VDU28,32,31,39,0
470 VDU24,0;0;1023;1023;
480 ENDPROC
500 DEFPROCgetvalues
510 VDU12,23;10,32,0;0;0;
520 REPEAT
530 PRINTTAB(0,12)"PRESS""WHITE""BUTTON""TO STARTOUTPUT"
540 VDU30
550 FZ=ADVAL(1)/SZ-100
560 PRINT"1st",FZ
570 H3Z=ADVAL(2)/SZ-100
580 PRINT"3rd",H3Z
590 H5Z=ADVAL(3)/SZ-100
600 PRINT"5th",H5Z
610 H7Z=ADVAL(4)/SZ-100
620 PRINT"7th",H7Z
630 UNTIL (?&FE40 AND 32)=0
640 VDU30
650 PRINTTAB(0,12)"PRESS""RED""BUTTON""TO STOP "
660 ENDPROC
670
680 DEFPROCplot
690 CLG
700 MOVE 0,512
710 DRAW 1279,512
720 FOR NZ=0 TO 255
730 XZ=INT (FZ*comp(0,NZ)+H3Z*comp(1,NZ)+H5Z*comp(2,NZ)+H7Z*comp(3,NZ))+128
740 IF XZ>255 XZ=255
750 IF XZ<0 XZ=0
760 ?(dataZ+NZ)=XZ
770 DRAW 4*NZ,XZ*4
780 NEXT
790 ENDPROC

```



# The Commodore 64



by Mike James

In the early days of personal computers there were two landmarks in machine design – the APPLE II and the Commodore PET. The APPLE II was important because it was the first machine to offer high resolution colour graphics. The PET was important because it introduced a completely self-contained machine, including the monitor and it introduced the use of memory-mapped block graphics. Both machines sold well and now APPLE and PET are part of the computer 'establishment'. However it is difficult to follow a successful machine with yet another success because of the pressures to stay compatible with your past machines. Commodore have attempted to maintain their market lead by producing improved versions of the familiar PET, often referred to as the 'SUPER PET' and by introducing new products such as the VIC 20 to cover areas of the market, such as game playing that the original PET format is unsuitable for. This is a very sensible way to try to keep the initial advantage of the PET and break new ground. However reviewing the current state of Commodore's market strategy leaves my head spinning with the names of so many products. I have now given up trying to see the sense in one computer company producing so many different machines and also talking about producing even more! For example as well as the PET 4000 series, the PET 8000 series, the 9000, the 500 series, the 700 series, the VIC 10, the VIC 20 and the VIC 30 we have the Commodore 64, the subject of this review! With so many machines already available and yet others rumoured it must cast doubt on the long term prospects for any one of them. Indeed rumours of the imminent replacement of the VIC 20 have been around for almost as long as the machine! The Commodore 64 is a machine that is being offered to two sections of the market – to the games playing community it is presented as an improved VIC 20 and to the business community it is being presented as a smaller cheaper PET or series 500 machine. If you look at the characteristics of the Commodore 64 then you can see how the 'confusion' arises. With a 40 column colour display and good sound facilities it should be good for games. However if you add a Z80 card and CP/M to it it could be a candidate for more serious computing. To find out exactly where the Commodore 64 fits into the scheme of things clearly requires a careful examination.

## The Machine

The Commodore 64 is superficially like the VIC 20. It certainly fits inside a plastic case that is the same size as the VIC 20's and uses the same keyboard but its overall specifications are much better. The only external detail worth commenting on is the use of a separate power supply. As Commodore owns the chip manufacturing company MOS Technology any machines that it makes can include chips designed specially and so it is with the Commodore 64. Inside you will find a special version of the well known 6502 microprocessor designated the 6510, two video processor chips the 6566 and the 6567 and a sound generator chip designated the 6581. With all this special electronics inside the 64 you might expect it to be different from the run of the mill machines built from off the shelf chips. And indeed to a certain extent this is the case. The 6510 contains some extra output lines that are used to control more than the usual 64K of memory. In fact the 6510 is a 6502 microprocessor coupled with a PIA (Peripheral Interface Adaptor) in the same package. As mentioned the output lines of the PIA are used to control memory by bank switching. The 64 in the Commodore 64's name comes from the fact that it contains a full 64K of RAM. However while running BASIC only about 38K of RAM is available, the rest of the memory space is taken up by 20K of BASIC ROM. If you don't need BASIC then the ROM can be switched out and more of the RAM used. Apart from this ability to reconfigure the memory the 6510 seems to be a fairly standard 6502 and capable of running 6502 machine code without any modifications. This of course is a good idea seeing that all the other Commodore machines run 6502 machine code.

The two video chips combine to provide a strange combination of advanced and simple graphics features. The first thing to say about the TV display is that it is an improvement over the VIC 20's miserable 20 characters to a line but is still only 40 characters to a line. Commodore are almost alone in believing that a serious machine can get by with a 40 column line – even APPLE have made provision for an 80 column card in their new APPLE IIE machine. A forty column screen is adequate for games but it makes serious applications such as word processors or 'spread sheet' calculations unnecessarily difficult. On the plus side the display does make available 16 colours, both upper and lower case characters and the well known PET style block graphics characters. The presence of the block graphics characters will please programmers used to the PET but what about high resolution graphics? There is supposed to be a 320 by 200 dot high resolution graphics mode but as there is no mention of how to use it in the manual provided with the machine I have to assume that it is for 'experts only'. The screen display is memory-mapped in two parts. The first part is used to store the ASCII codes of the characters, one memory location to each screen location. The second screen area similarly has one memory location per screen location but in this case it is used to store the colour code of the character. (This means that the Commodore 64 controls its displays using parallel attributes – for a discussion of parallel and serial attributes see the Oric review in last month's E & CM). An advanced feature that is



explained in the manual is the provision of high resolution 'sprites'. A sprite is a fairly recent invention and consists of a user-definable graphics character that is associated with a position on the screen. Unlike ordinary graphics characters you do not have to print a sprite to make it appear on the screen it is always present (while enabled) and moves around as two memory locations holding the X and Y co-ordinates. Notice that there is no need to blank out the old image of the sprite by printing blanks and this means that a sprite can be moved around the screen as fast as two memory locations can be changed. The importance of this is that a number of objects can be animated on the screen even from BASIC. The Commodore 64 is not the only machine that has sprites but its eight fairly large high resolution sprites is difficult to beat.

The sound chip is also quite something. It has three tone and one noise channel, working over nine octaves with a limited form of envelope control. The sound is added to the UHF modulated signal and produced through the TV's own loudspeaker – a method that is usually better than building a loud- (or fairly soft in some cases!) speaker into the computer.

However the new Commodore device is not that different from the well known AY-9-8910 sound generator chip and the new device is really only new to Commodore! From the point of view of pure sophistication in sound generation the BBC Micro still has the lead but not because of better sound generating hardware but because of the software that uses it.

### The Software

The Commodore 64 uses the early version of Microsoft BASIC that has become associated with the PET and for this reason is often known as PET BASIC. To be more precise it is the same version of BASIC as used on the VIC 20. There is a lot to be said for sticking with a well-known version of BASIC but even Microsoft BASIC has moved on since the PET and now includes extra features that makes it more attractive and newer BASICs such as BBC BASIC are even more advanced. Seen against this background the BASIC on the 64 is primitive. What is even worse is that no extra facilities to control the out of the ordinary hardware is included. To control the video screen, sprites and sound generator you have to use PEEKs and POKEs in large numbers. Other computers have added commands such as SOUND, PLAY, DRAW etc. to their BASICs to make programs easier to write. This certainly makes their version of BASIC non-standard but to adopt the opposite view and try to handle all the extra hardware via standard PEEKs and POKEs is ridiculous. To define a sprite or to make a sound take an uncomfortable number of PEEKs and POKEs and this makes the possibility of error much greater. To use PEEKs and POKEs to control special hardware is one thing but to have to POKE graphics characters and colour codes into the screen memory map to produce shapes at an exact X, Y location is very laborious.

The Commodore 64 uses the standard PET cursor control characters to move the current printing position around the screen but there is no equivalent to the TAB(X,Y) function to move the current printing position to any point on the screen.

Similarly colour control codes can be quoted in print statements but if you want to alter the colour of a particular point then once again you have to resort to POKEs.

The lack of specific commands to control the Commodore 64 leaves the programmer with a strange feeling of using a BASIC that was never intended for the machine. There are undeniable advantages to using a PET like BASIC but if the Commodore 64 is intended to be used by novice programmers to create games etc. then special commands to handle the exciting hardware inside the machine. It seems a pity that after providing the hardware to make it possible to move coloured shapes around the screen at a reasonable speed using nothing but BASIC, and so making it possible for novice programmers to do things that previously would have needed machine code, the novice programmer is forced to learn how to handle binary numbers and memory locations!

### Expansion

The Commodore 64 needs a TV set either colour or black and white to be of any use at all. However if you are going to actually get any value out of the machine you must also have some means of loading and perhaps saving programs. There are three ways in which this can be done – cassette, cartridge or disc. Cassette is obviously cheaper than disc for program storage but in the Commodore 64's case you have to use one of the specially made tape drives costing around fifty pounds. This may come as something of a shock to anyone expecting to use an existing tape recorder with the machine. You can also use ROM cartridges containing pre-packaged programs instead of writing your own but the cartridge slot on the 64 is a different size to that of the VIC 20 and in any case all of the programs for the VIC would have to be re-written to take account of the hardware differences. The final and most expensive option is to use a disc drive. Once again you won't find a bargain in that a single drive costs around three hundred pounds but you will find a nicely engineered product based on a half height five inch drive.

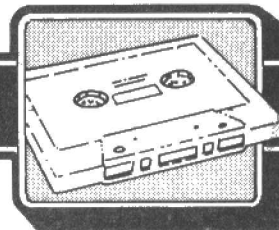
To fit its image as a games machine the 64 has two joystick ports. However in line with its image as a serious machine you can buy an IEEE 488 interface cartridge that will give you access to the wide range of PET peripherals and a Z80 card that will allow you to use CP/M and a wide range of business software.

### Documentation

Commodore have long been known as the arch villains of the incomprehensible computer manual that the Commodore 64's manual came as a pleasant surprise. The user manual supplied with the machine is well produced and fairly readable. It is aimed at the beginner and so it lacks any real technical information. For the more technical minded programmer there is a more advanced manual but you have to buy it as an extra. This division of information into two manuals is very praiseworthy as far too many computer manuals try to address themselves to too wide an audience. However there is

developed hardware and underdeveloped software. With a forty column screen, sprites and joystick ports the machine is well suited for game playing. However at £300 it is an expensive games machine. With its expansion possibilities the machine could form the basis for a low cost business system but with a forty column screen it is forever limited. The best description of the Commodore 64 is as an expensive games machine that offers the possibility of some serious use. Without a good extended BASIC the complete novice will be better off using package software or not using all of the advanced hardware within the machine. In short the Commodore 64 is a machine in search of a good language.

# Road Runner®



# DRAGON SOFTWARE

*Reviewed by S. M. Gee  
and Kay Ewbank*

The Dragon arrived in the shops only last September so there has only been six months for many of the software houses to produce programs for it. At first there was a marked lack of software for the Dragon but now the situation is very different. In terms of the number of titles the amount of Dragon software is quite remarkable given the relatively short time period for its development. However, despite the large number of titles, closer examination reveals that there is not as great a variety of programs as we might have expected. A great number of the programs for the Dragon appear to be variations on the same theme. In particular, there are lots of wordy adventure games including three really similar ones from Dragon Data. We also received five versions of Golf to review for this feature!

Having received a total of 69 cassettes/cartridges from 17 different software firms we were unable to run and evaluate every single program. We therefore decided to look at at least one tape from each of the producers and try to indicate the range of the choice available.

Only two manufacturers, Compusense and Dragon Data, sent us cartridges as well as cassettes. Given that its cartridge slot is one of the special features of the Dragon we found this rather surprising and also rather disappointing as in general the cartridge games we played were much more exciting than those on cassettes.

If you have tried to write your own programs for your Dragon you will probably appreciate some problems experienced by the software producers—to mention just two, low resolution graphics are difficult to program and there are restrictions on the colour combinations you can use. Moreover, Dragon BASIC is relatively slow. This

problem can of course be overcome by writing programs in machine code but this option has in general not been used with the result that many games do run slowly and many are unresponsive—a particular problem in the case of the games that use joysticks. The exception to this reliance on BASIC is Dragon Data's cartridges which are written in machine code and therefore run very much faster.

We had very few problems in loading any of the software we were sent and all of them were successfully loaded. Many of the manufacturers had taken the precaution of saving their programs more than once on the tapes supplied but this seemed to be unnecessary.

**BESERK (cartridge) (£19.95)  
ASTROBLAST (cartridge)  
(£19.95)  
COMPUTAVOICE (£8.00)  
and CALIXTO ISLAND  
(£8.00)  
from DRAGON DATA LTD.,  
Kenfig Industrial Estate,  
Margam, Port Talbot,  
SA13 2PE**

In general we were very impressed by Dragon Data's cartridge software, all of which was fast-moving and very responsive to the user's input. In BESERK you control a little man figure who can move through a series of rooms and the object of the game is to destroy hostile robots. The game is made more difficult by the presence of Evil Orville who bounces through walls and demolishes everything in his path—including you. We found this a difficult game to win and wished that there was an easy level at which to play it!

ASTROBLAST is an attractive variation on an invaders type game. Played with joysticks it is fast moving and responsive and its tiny white stars and shimmering spaceships on a black background combine to give a convincing display.

The Dragon has not got a resident speech synthesiser so we were interested in hearing the sounds produced by COMPUTAVOICE. Its demonstration words are the numbers one to nine and they were reasonably recognisable even if rather unpleasing to the ear. Programming in your own words is a matter of splitting them into their phonemes. After a little experimentation it is possible to produce some simple words but as with any sounds produced this way the results lack any naturalness and so have little practical application.

CALIXTO ISLAND is one of the words only adventure games that seem very popular on the Dragon. Each time we played it we found our starting point was Dr. Lagarto's study and we have now found a number of equally interesting locations. One frustrating aspect of this game was trying to obtain an inventory of objects being carried. Although the instructions suggested that the Dragon would understand the noun "inventory" on its own it patently did not and we tried verbs like list and give before eventually we hit upon "take inventory" which worked! Once you really get into this game it is really compulsive but it is easy to get put off by an initial lack of knowledge of its commands.

**MAZE RUNNER (£3.00) and  
DESIGNER (£8.00) from  
QUODLIBET, 2 Victoria  
Terrace, Dorchester, Dorset,  
DT1 1LS.**

When you run MAZE RUNNER a three dimensional space is displayed on the screen in high resolution graphics. The object of the game is to escape from the maze, via a door which is locked, before your air supply is exhausted. There are two components to the game—moving through the maze and cracking the colour code to unlock the door. Although impressed by the graphics display which gives a convincing impression of moving through a 3D



space, we did not find this game particularly compulsive. However, given that its recommended price is only £3.00 it seems good value.

Quodlibet's Designer program allows you to produce graphics characters of your own design more easily than if you were to start from scratch. In other words, it is a tool that enables you to create graphics displays on your Dragon. Although the instructions that accompany the program are very brief, it is fairly easy to use to design simple shapes.

## **STRATEGIC COMMAND (£9.95) from ROMIK, 24 Church Street, Slough SL1 1PT**

This is an action graphics game for two players. The object of the game is to dominate the world. Each player has to define his or her own capital and bases and attack the other player's and to do so has a variety of land and sea forces. The game is played using joysticks and a number of different actions are required since there are a number of options available. To play the game properly you need to understand the details of the rules. These are given in extremely small print on the cassette library case inlay. No instructions at all are given within the game which we thought rather a pity.

## **CITY DEFENCE (£5.75) from SHARDS SOFTWARE, No. 10, Park Vale Court, Vine Way, Brentwood CM14 4UR**

This is an implementation of a fairly well known arcade game in which you have to defend six cities which are being attacked by encroaching beams of enemy fire. You have to position your cursor at the very end of each of the "streamers" using joystick controls and then fire. This game employs two alternative colour combinations – the second of which flashes on which one of your cities is hit – and makes good use of a variety of sound effects, however because it is written in BASIC it is rather unresponsive and the user has insufficient control both of movement and firing.

## **QUEST (£5.00) from IMPACT, 70 Redford Avenue, Edinburgh EH13 0BW.**

This is a fairly typical adventure game in which you are trying to find the Holy Grail hidden within a maze of chambers. You can move in the four compass directions, you can look into the adjacent chambers, you have to fight off the monsters that you meet keeping an eye on your six requirements for strength, stamina, intelligence, dexterity, food and drink. This is an enjoyable and challenging adventure which seems good value for money.

## **ST. GEORGE AND THE DRAGON (£6.95) from COMPUTER RENTALS LTD., 140 Whitechapel Road, London E1.**

This is a graphics game written in BASIC and played using the joysticks. We found we needed to plug into the opposite joystick port from that suggested so if you also experience this hitch don't despair just try the other port. The problem with this game was that the joystick controls were not sufficiently responsive which meant that the hazards were even worse than intended. Moreover, having killed the dragon and crossed the slippery bridge we found that the magic stone refused utterly to be smote – with the result that we never discovered what happened when the maiden was released – this was indeed a disappointing anticlimax!

## **GOLF (£4.95) from APEX TRADING COMPANY, 115 Crescent Drive South, Brighton BN2 6SB.**

The graphics used for this program are very pleasing and the player has a choice of three factors – club strength, strength of shot and the angle of the shot – which add to the interest of the game. Unfortunately the ball moves very slowly indeed which makes the game unnecessarily boring – it would have been better if fewer points on the ball's trajectory were displayed. When you get your ball onto the green there is only one decision to make for each shot – the club

strength. A "real" golfer may find this game confusing as it numbers the irons the wrong way round. For future reference, a nine iron is the lightest, not the heaviest!

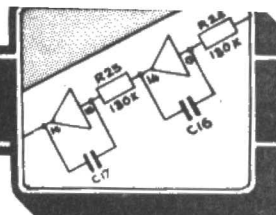
## **SCARFMAN (£8.00) from MICRODEAL, 41 Truro Road, St. Austell, Cornwall PL25 5JE**

This is a fast moving and colourful implementation of the well-known Pacman game. You can play either with joysticks or via the keyboard. The object of the game is to survive and to eat the blue monsters before the red monsters eat you! Every time you eat a blue monster it reappears as a red monster and every time you eat a cross all the red monsters turn blue. As you get more successful the monsters turn red quicker so the game gets harder. In the long run you can't win but just keeping going is hilarious fun.

## **EDUQUIZ 1 (£9.95) from GEM SOFTWARE, Unit D, The Maltings, Station Road, Sawbridgeworth, Herts.**

GEM SOFTWARE'S catalogue includes two general knowledge tapes and we tried one of them. It contained a geography quiz (about the rivers, mountains and capitals of countries of all continents), one about inventors, (requiring knowledge of their inventions, their countries of origin and the dates of their inventions) and one about the monarchs of England (when their reigns started, significant events during their reigns and who they were married to). Each quiz is prefaced by a graphics display and a snatch of music but no graphics are required during each twenty question quiz. Sound is used to indicate right or wrong answers. The method of presentation – multiple answer questions plus a betting routine – certainly add to the entertainment value of these sets of quizzes and the way that new questions are related to but not identical to previous tests means that you can improve your knowledge by playing repeatedly.

**\*All prices are inclusive of VAT, postage and packing.**



# Software For The System

by John Chewter

## Part 7 of the E & CM Home Computer Project

This month we will look at driving the board from basic. The principles are the same as using machine code but is, of course, slower.

The program was written in T.S.C. Basic and has no monitor program calls at all. It has been written this way to allow the board to be tested and demonstrated using S.Bug, etc.

Hi Bug etc. will contain, plot, move and draw facilities but these are not used in the programme but will be discussed later.

### Demonstration Programme

It is not expected that the user will enter all the REM statements (!) but these have been liberally used to aid following the programme.

Lines 40-74 assign values to variables and initialise the video chip. Control is then passed to line 3000.

The first part of the demonstration writes E & CM - 1024 times, each time in a different size, shape or axis. The colours of border, background and letters alter as the program progresses. The second part now starts by clearing the screen and draws a pattern (STARBURST I) reminiscent of an explosion in a paint factory. This is drawn on page one and when completed a similar pattern, but with a shifted focus (STARBURST II) is drawn on page 2.

Lines 3100-3170 then alternate the display pages one and two to produce an interesting affect.

Lines 300 to 490 comprise a (long-winded) plot routine. This has great scope for optimising, but has been done this way to illustrate the principle, (a machine code version of this will be in Hi Bug). When called it draws a line from the present position to the point X,Y.

Because the Delta registers are only single byte and the screen is 512 x 512, it is necessary to draw two lines for a line with an offset greater than 256 in the X or (and) Y direction.

We cheat here by always assuming that we have a line bigger than 256 bits offset.

The idea is to calculate the offsets from present position to new position, to place half of these values into the delta X and delta Y registers, to draw a line to a position half way to the destination point and then to place the remainders in the respective registers, to complete the line. This routine is used in the sketch-pad and goldfish routines. After the starburst demonstrations, the screen clears to green and the user is invited to enter the X,Y end point and colour of the line which is then drawn. Pushing any non-numeric key will cause an exit from the routine and the electronic goldfish routine commences.

The goldfish is drawn in lines 5020 to 5200. An interesting feature here is the short vectors in 5095 and 5100 (this draws the fish's mouth). These single commands contain all the information to draw short vectors i.e. and direction information and does not use the Delta X and Delta Y registers.

The fish (white) wanders around the tank (blue) in a pseudo-random fashion, leaving a trail of images of itself in black. It is recommended that potential board users should become familiar with the above programme and the following points investigated.

The movement of the goldfish is accomplished by using the plot routine (line 6065) to avoid a line being drawn the screen is white - protected using a poke CM, 3 command i.e. this command lifts the "pen" off the paper (Poke CM,2 reverses this).

To make the fish swim about without leaving a trail change 6060 to Gosub 1000: Poke CL, 1 which will over-draw the last fish plot with one in blue i.e. background colour. Poking (CM + 2) with a number between 0 and 3 inclusive (line 4000) will produce variations on the starburst theme because dotted, dashed, dotted and dashed or plain lines will be drawn, to complete the pattern.

Line 1000 is very important. Because the board takes a finite time to do things especially clear the screen etc. it is necessary to check to see if the chip has completed the last task before issuing a new instruction or changing the colour register. E.g. removing line 110 will clear the screen to the foreground colour and this the routine will be writing in blue on blue or white on white etc. When writing in basic we have found that many of the "Gosub 1000 lines can be removed (especially if a multiply or divide has just been executed).

These statements take longer to execute than the chip takes to execute the last instruction. It has been found to be good policy to put all of these routines in, initially, and to run the programme successfully before removing these checks, one at a time. It is always necessary to use checks when using assembler.

Line 2000 is a variation on the line 1000 theme but it first waits until the display is in the vertical blanking phase. This approach can "clean up" animation routines which clear the screen before drawing the next image (because most of the clearing occurs off screen).

Using the Hi-Bug and HT-Bug rom based routines is much easier than the above and will briefly be introduced here.

Because of the commercial problems of altering somebody else's Basic we have a slightly unusual approach. The output routines check to see if the character is a "\$15 (SI). If it is, it switches to another mode and checks to see if the next character is a "P" (for plot a point) a "D" (for draw a line) or an "M" (move to a new position). New values for X and Y are now passed to the calculating routines which execute the graphics instructions. Not as painful as it seems. To use this method at the beginning of the program enter the following:

```
10 P$ = CHR(15) + "P"
11 M$ = CHR(15) + "M"
12 D$ = CHR(15) + "D"
```

To draw a line from a new position A,B to C, D. All the "busy" checks are done in the monitor program.

```
15 Print M$;A;B
20 Print D$;C;D
25 End
```


### Next Month

Floppy Disc Card.

```

1 REM *****
10 REM ***** HI-RES DEMO PROG. *****
11 REM *****
12 REM
13 REM THIS PROGRAM DEMONSTRATES THE USE OF THE EACH HI-RES.
14 REM VIDEO BOARD USING BASIC ROUTINES ONLY.
15 REM NO MONITOR ROUTINES ARE USED
16 REM THE PROGRAM IS NOT OPTIMISED, BUT IS WRITTEN TO ILLUSTRATE
17 REM THE PRINCIPLES INVOLVED.
18 REM
19 REM
20 REM ***** SET UP VARIABLES *****
21 B=0:C=4
22 CH=HEX("F100") REM COMMAND/STATUS REGISTER
23 M=CH+8:XL=CH+9 REM ADDRESS OF X REG./XL=LSB
24 Y=CH+10:YL=CH+11 REM DITTO FOR Y
25 D=CH+5:D1=CH+7 REM DELTA X AND DELTA Y REGS.
26 S2=CH+3 REM CHARACTER SIZE REG.
27 CL=HEX("F120") REM COLOUR REG. ADDRESS
28 POKE CH,6
29 POKE (CH+1),3:POKE (CH+2),8 REM INITIALISE REGISTERS
30 GOTO 3000 REM GO TO START OF MAIN PROGRAMME
31 REM
32 REM ***** THIS DISPLAYS 'EACH' IN 256 DIFFERENT SIZES *****
33 REM ***** IN 4 MODES WITH CHANGING COLOURS ETC. *****
34 FOR X=0 TO 255
35 POKE (CH+2),8 REM SET CTRL2 REG. FOR NORMAL CHARACTERS
36 POKE S2,X REM SET SIZE/SHAPE WITH X
37 FOR G=1 TO 2
38 GOSUB 1000 REM BOARD NOT READY?
39 POKE CL,(A+3) REM SET COLOUR - BACKGROUND
40 POKE CH,6 REM CLEAR PAGE 1
41 GOSUB 1000 REM FINISHED?
42 POKE CL,A+1 REM SET FOREGROUND COLOUR
43 GOSUB 1000 REM BUSY
44 REM
45 REM ***** THIS OUTPUTS 'EACH' *****
46 POKE CH,69
47 GOSUB 1000
48 POKE CH,33
49 GOSUB 1000
50 POKE CH,67
51 GOSUB 1000
52 POKE CH,77
53 GOSUB 1000
54 A=A+8 REM CHANGES COLOUR COMBINATION
55 POKE HEX("E000"),A REM SETS BORDER COLOUR
56 POKE CH+2,C REM CHANGES CHARS. TO ITALICS
57 NEXT G REM REPEATS CHAR. BUT IN ITALICS
58 NEXT X REM NEXT SIZE
59 RETURN REM END OF SUBROUTINE
60 REM
61 REM ***** PLOT SUBROUTINE *****
62 REM DRAWS A LINE FROM PRESENT POSITION TO A POINT X,Y
63 REM DONE IN 2 HALVES AS DELTA REGS. ARE ONLY 256 IN SIZE
64 REM NEXT LINE CALCULATES SIGNED OFFSET FOR X
65 D=X-(PEEK(XL)+255*(PEEK(YL)))
66 N=ABS(D) REM FOR X VALUE
67 H=INT(N/2) REM HALF OFFSET
68 POKE D,H REM LOAD DELTA X REG WITH UNSIGNED OFFSET
69 E=Y-(PEEK(YL)+255*(PEEK(YH))) REM OFFSET FOR Y
70 N=ABS(E)
71 H=INT(N/2)
72 POKE D,H REM LOAD UNSIGNED OFFSET INTO DELTA Y REG
73 REM SIGNS OF THE X & Y OFFSETS FORM PART OF DRAW COMMAND
74 SX=SGN(D):SY=SGN(E)
75 ON (SX+2) GOTO 420,430,430
76 IF SY=0 THEN CX=19 ELSE CX=23
77 GOTO 440
78 IF SY=0 THEN CY=17 ELSE CY=21
79 GOSUB 1000 REM CX=DRAW CMD. (INCLUDES DIRECTION)
80 POKE CH,N+H REM REPEAT FOR 2ND HALF OF LINE
81 POKE CH,N+H
82 GOSUB 1000
83 POKE CH,CX
84 RETURN REM END OF PLOT SUBROUTINE
85 REM
86 REM *** THIS ENSURES THAT LAST INSTRUCTION HAS FINISHED ***
87 IF (PEEK(CH) AND 4) > 0 THEN RETURN ELSE GOTO 1000
88 REM
89 REM
90 REM *** CHECKS TO SEE IF FINISHED AND IF VERT. BLK IS OCCURRING ***
91 IF (PEEK(CH) AND 6) > 0 THEN RETURN ELSE GOTO 2000
92 REM
93 REM
94 REM ***** START OF MAIN PROGRAMME *****
95 GOSUB 80 REM PRINT 'EACH' (HORIZ.) ROUTINE
96 B=0 REM CHANGES DISPLAY MODE TO VERTICAL
97 C=12 REM CHANGES MODE TO VERT. ITALICS
98 GOSUB 80 REM PRINT 'EACH' VERT. ITALIC
99 REM PREPARE FOR 'STARBURST' 11 DISPLAY
100 T=0 REM POSITIONS FOCUS OF DRAWING TO CENTRE
101 POKE (CH+2),8 REM SET UP FOR NORMAL LINE DRAWING
102 GOSUB 4000 REM DRAWS PICTURE 1
103 REM PREPARE FOR 'STARBURST' 11 DISPLAY
1040 T=127 REM SETS FOCUS OF DISPLAY TO TOP RIGHT
1050 POKE CH,0 REM CHANGE PAGE
1060 REM
1070 REM ***** ALTERNATES PAGES 1 AND 2 TO PRODUCE MOVING DISPLAY **
1080 FOR K=1 TO 25
1090 POKE CH,1 REM DISPLAY PAGE 1
1100 GOSUB 3500 REM DELAY ROUTINE
1110 POKE CH,0 REM DISPLAY PAGE 2
1120 GOSUB 3500
1130 NEXT K
1140 REM
1150 REM ***** SKETCH PAD ROUTINE *****
1160 ON ERROR GOTO 10000 REM TRAPS 'NON-NUMBERS'
1170 POKE CL,2:POKE CH,12 REM CLEAR SCREEN TO GREEN
1180 PRINT "***** SKETCH PAD *****"
1190 PRINT "TYPE ANY LETTER TO QUIT"
1200 INPUT "VALUE OF X",X
1210 IF X>510 THEN GOTO 3200
1220 IF X<0 THEN GOTO 3200
1230 INPUT "VALUE OF Y",Y
1240 IF Y>510 THEN GOTO 3200
1250 IF Y<0 THEN GOTO 3200
1260 INPUT "COLOUR (R=4,G=2,B=1,YL=6,CY=3,M=5,W=7,BLK=0)",Z
1270 IF Z<0 THEN GOTO 3240
1280 IF Z>7 THEN GOTO 3240
1290 POKE CL,2:GOSUB 1000 REM LOADS COLOUR OF LINE
1300 GOSUB 300 REM PLOT LINE
1310 GOTO 3200 REM NEW LINE
1320 REM
1330 REM ***** DELAY ROUTINE *****
1340 FOR L=1 TO 200
1350 M=1
1360 NEXT L
1370 RETURN
1380 END
1390 REM
1400 REM ***** STARBURST PATTERN GENERATOR *****
1410 REM THIS PLOTS PLAIN LINES TO VARY CHANGE '4000' TO POKE A VALUE
1420 REM BETWEEN 0 AND 3, INC.
1430 POKE CH+2,0
1440 POKE HEX("E000"),4 REM SET BORDER TO RED
1450 POKE CH,5 REM HOME CURSOR
1460 POKE CL,0:POKE CH,12 REM CLEAR SCREEN TO BLACK
1470 GOSUB 1000 REM FINISHED?
1480 FOR Y=0 TO 510 STEP 8
1490 FOR X=0 TO 510 STEP 16
1500 POKE XL,Y:POKE YL,X REM SET FOCUS TO RIGHT PLACE
1510 POKE XM,1
1520 POKE YH,1
1530 POKE CL,Y/7 REM MODIFY COLOUR
1540 GOSUB 300 REM PLOT LINE
1550 NEXT Y
1560 NEXT X
1570 GOSUB 1000 REM END OF ROUTINE
1580 RETURN
1590 REM
1600 REM ***** THIS ROUTINE DRAWS A GOLDFISH AT X,Y *****
1610 POKE DX,5:POKE DY,10 REM SETS PROPORTIONS OF FISH
1620 POKE CH,17:GOSUB 1000 REM DRAW COMMANDS
1630 POKE CH,17:GOSUB 1000
1640 POKE CH,21:GOSUB 1000
1650 POKE CH,23:GOSUB 1000 REM SHORT VECTOR!!!!!!
1660 POKE CH,23:GOSUB 1000 REM DITTO
1670 POKE CH,19:GOSUB 1000
1680 POKE CH,19:GOSUB 1000
1690 POKE CH,20:GOSUB 1000
1700 POKE CH,20:GOSUB 1000
1710 RETURN REM END OF SUBROUTINE
1720 REM
1730 REM ***** ELECTRONIC GOLDFISH IN BOWL *****
1740 REM FISH WANDERS ABOUT IN A BLUE TANK, LEAVING A TRAIL OF OUTLINES
1750 REM IN BLACK TO SHOW WHERE HE'S BEEN
1760 POKE HEX("E000"),4
1770 POKE CL,3:POKE CH,12 REM CLEAR SCREEN TO BLUE
1780 POKE CH,5 REM HOME CURSOR
1790 GOSUB 1000:POKE CL,7 REM WHEN FINISHED SET TO WHITE
1800 GOSUB 5000 REM DRAW FISH AT X,Y
1810 F=10 AND (010) G=11 AND (0)
1820 IF F=1 THEN I=0
1830 IF I=0 THEN I=1
1840 IF I=1 THEN I=0
1850 IF F=1 THEN H=0
1860 IF H=0 THEN H=1
1870 IF G=0 AND (G/5) THEN H=1
1880 IF H=1 THEN G=0
1890 IF I=1 THEN F=F+1
1900 H=ABS(H)+F:RES=INT(W) REM COMPUTE NEW X & Y POSITIONS
1910 GOSUB 1000:POKE CL,2 REM CHANGE COLOUR TO BLACK
1920 GOSUB 5000 REM DRAW FISH
1930 POKE CL,7 REM WHITE PROTECT SCREEN RAM
1940 GOSUB 300 REM PLOT TO NEW POSITION
1950 POKE CH,0 REM ERASE SCREEN RAM
1960 GOTO 6210 REM REPEAT
1970 REM
1980 REM ***** END ROUTINE FOR SKETCH PAD *****
1990 PRINT "THAT'S ENOUGH" IF OF NON-NUM
2000 IF A=1 THEN GOTO 6000 ELSE RESUME

```



"THERE MUST BE A  
COMPUTER DEALER I CAN  
TURN TO FOR  
GUIDANCE...."

If you're still staggering through the computer jungle and not getting sensible answers to your questions, we have some good news:

Now you can turn to **professional** people who are capable of giving you sound advice on practically all aspects of popular computing.

They all have one thing in common:

They are **COMPUTERS FOR ALL** dealers.

A Computers for All dealer is different from the normal Computer retailer.

Not surprisingly he won't try to sell you things like cameras or cosmetics; stationery or sealing wax.

He will, however, be capable of answering sensibly almost any question you have on computers and computing, and have readily available a wide range of popular computers, hardware, software, books, and peripherals.

So why not call in at your local **COMPUTERS FOR ALL** dealer today?

He can lead you in the direction you want to travel.

### The shops where people matter:

**Akhter Instruments**, Unit 19, Arlinghyde Estates, South Road, Harlow, Essex. 0279 412639. **Anirog Computers**, 26 Balcombe Gardens, Horley, Surrey. 02934 6083. **Aphros Software Co.**, 83 Canterbury Road, Westbrook, Margate, Kent. 0843 23627. **Automation Services (S. Wales)**, 3 Wermey Road, Penysfal, Bridgend, S. Wales. 0656 720959. **Bits & Bytes**, 44 Fore Street, Ilfracombe, N. Devon EX34 9JD. 0271 62801. **Carlton Computers**, 4 Swanstons Road, Great Yarmouth, Norfolk NR30 3NQ. 0493 58898. **Computasolve**, 8 Central Parade, St. Marks Hill, Surbiton, Surrey. 01-390 5135. **Computer Systems (Torbay)**, Pump Street, Brixham, Devon. 08045 6565/6. **Computers for All**, 72 North Street, Romford, Essex. 0708 752862. **Crystal Computers**, 209 Union Street, Torquay, Devon. 0803 22699. **Dan Evans (Barry) Ltd.**, 81 Holton Road, Barry, South Glamorgan. 0446 734242. **Death Valley Computers**, P.O. Box 54, Worcester WR2 6QA. 0905 640400. **D. V. Martin Ltd.**, 13 Bridge Street, Belfast, N. Ireland BT1 1LT. 0232 226434. **Emprise Ltd.**, 58 East Street, Colchester, Essex. 0206 870353. **Emprise Ltd.**, 3a Baddow Road, Chelmsford, Essex. 0245 356834. **Eurocalc**, 224 Tottenham Court Rd., London W1. 01-631 4139. **Evesham Micro Centre**, Crown Court Yard, Bridge St., Evesham, Worcester. 0386-48635. **Fal-Soft Computers**, 8 St. George's Arcade, Falmouth, Cornwall. 0326 314663. **Home Computer Centre (Ross Records)**, 13 Kingston Road, Portsmouth, Hampshire. 0703 819515. **Impulse Micro Systems Ltd.**, 6 Central Chambers, Cooks Alley, Wood Street, Stratford-upon-Avon. 0789 295819. **Jade Computers**, Coombend, Radstock, Bath, Avon. 0761 32570. **Kelly's Computermarket**, 227 Dartmouth Road, Sydenham, London SE26 4QY. 01-699 4399/6202. **Kenneth Ward Computers**, Verve House, London Road, Sunningdale, Berkshire. 0990 22275. **Medway Computers**, 141 New Road, Chatham, Kent. 0634 826080. **Mercator Computer Systems**, 3 Whiteladies Road, Clifton, Bristol. 0272 731079. **Micro Magic**, 128 Erith Road, Bexleyheath, Kent. 0322 523052. **Mobile Micros**, 2 Castle Street, Thornbury, Bristol. 0454 418383. **Morrison Computer Centre**, 37 Clase Road, Morriston, Swansea SA6 8DS. 0792 797572. **Steve's Computer Co. Ltd.**, Castle Arcade, Cardiff. 0222 41905. **The Computer Centre (BMS) Ltd.**, 37d & 37e, Robertson Street, Hastings, East Sussex. 0424 439190. **Tomorrow's World**, Esplanade, Lerwick, Shetlands ZE1 0LL. 0595 2145. **Twilstar Computers**, 17 Regina Road, Southall, Middlesex. 01-574 5271. **Weytech Computer Systems**, 20 St. Edmunds Street, Weymouth, Dorset. 03057 79881.



**COMPUTERS  
FOR ALL**